
*A Microarchitectural Survey
of
Next Generation Microprocessors*

**AMD K5
DEC Alpha 21164
HP PA-8000
Intel P6
MIPS R10000
PowerPC 620
Sun UltraSPARC**

Ed Tam

April 25, 1995

Term Project Report

EECS 598.3

Winter Term, 1995

Table of Contents

List of Figures	iii
List of Tables	iii
Abstract	iv
Note on References	v
1.0 Introduction	1
2.0 Background	2
2.1 Intel P6	2
2.2 AMD K5	3
2.3 PowerPC 620	4
2.4 DEC Alpha 21164	4
2.5 MIPS R10000	5
2.6 Sun UltraSPARC	5
2.7 HP PA-8000	6
3.0 Predecode and Fetch	7
4.0 Pipeline Lengths	8
5.0 Decode	11
6.0 Branch Prediction	13
6.1 Prediction Algorithm	13
6.2 Accuracy	15
6.3 Compiler Support	15
6.4 Recovery	16
6.5 Number of Speculated Branches	17
6.6 Penalties	17
7.0 Register Renaming and Dependency Resolution	18
8.0 Dispatch	20

9.0 Issue	21
10.0 Functional Units	23
10.1 Integer Execution Units	23
10.2 Floating Point Execution Units	24
10.3 Load/Store Units	26
10.4 Other Functional Units	28
11.0 Caches	29
11.1 Instruction Cache	29
11.2 Data Cache	31
11.3 Level 2 (L2) Cache	32
12.0 Retirement	34
13.0 System Interface	36
14.0 Physical Characteristics	37
15.0 Processor Performance	38
16.0 Conclusion	41
Appendix: Processor Quick Reference Sheets	42
Advanced Micro Devices K5	43
Digital Equipment Corporation Alpha AXP 21164	44
Hewlett-Packard Precision Architecture 8000 (PA-8000)	45
Intel P6	46
MIPS Technologies, Incorporated R10000	47
Apple/IBM/Motorola PowerPC 620	48
Sun Microsystems UltraSPARC	49
References	50

List of Figures

Figure 1: PowerPC 620 pipeline	8
Figure 2: K5 pipeline	9
Figure 3: 21164 pipeline	9
Figure 4: UltraSPARC pipeline	9
Figure 5: R10000 pipeline	10
Figure 6: PA-8000	10
Figure 7: P6 pipeline	11
Figure 8: K5 and P6 decode process	12
Figure 9: Relative SPECint92 and SPECfp92 performance	39
Figure 10: SPECint92/MHz comparison	40

List of Tables

Table 1: Number of predecode bits added	7
Table 2: Branch prediction algorithm, implementation, and accuracy	15
Table 3: Penalties for branch prediction outcomes, in number of cycles	18
Table 4: Latencies for floating point operations, in cycles	25
Table 5: Instruction cache characteristics	29
Table 6: Data cache characteristics	31
Table 7: L2 cache characteristics	33
Table 8: Retirement characteristics	34
Table 9: System bus characteristics	36
Table 10: Physical characteristics	37
Table 11: SPEC92 benchmark performance	38

Abstract

The performance of microprocessors has been increasing at an impressive rate, doubling the performance of each previous generation approximately every 1.5 years. In late 1994 and early 1995, nearly every major microprocessor vendor has announced their “next generation” microprocessor. The processors include the AMD K5, the DEC Alpha 21164, the HP PA-8000, the Intel P6, the MIPS R10000, the PowerPC 620, and the Sun UltraSPARC. This report surveys the microarchitectural aspects of each of these planned microprocessors, comparing the microarchitectural decisions made by each design and the resulting performance each achieved. The report compares the implementations in an order following an instruction’s execution through the processor, starting from its fetch from off-chip memory into instruction caches, through decode, dispatch, issue, and execution, and finally retirement to architecturally visible state. Along the way, relevant parts of the microprocessor are discussed, including the decode process for the x86 CISC-based processors, the mechanisms used to buffer instructions stalled due to dependencies, functional unit characteristics, and cache designs. Finally, the processor’s interface to the outside world, the system interface, is discussed. An overview of each processor’s physical realization is presented, followed by a performance analysis of all the processors. Concluding remarks are then made about the current crop of microprocessors, with respect to their place in microprocessor evolution.

Note on References

A variety of sources of information was used to research each microprocessor. Due to the large amount of information that was common to each source for a given processor, no references are noted within the following report's text. However, at the end of the paper, in the References section, the references used are presented, grouped by microprocessor, for easy retrieval for further research into any given microprocessor. I felt that removing the numerous reference notes required for each processor aspect would ease the reading of the paper, especially considering the fact that this is a survey of microprocessors and does not present original work.

1.0 Introduction

The performance of microprocessors has been increasing at an impressive rate, doubling the performance of each previous generation approximately every 1.5 years. While it is important to consider the designs for processors in the near future, reviewing currently available, cutting edge technology can be quite revealing. By looking at the features incorporated by microprocessor designers in their current generation products, we can see what was deemed feasible, given their cost restrictions and target markets, to be built into a microprocessor. Many of these features were proposed earlier in academia; seeing their utilization in actual shipping microprocessors is an encouraging prospect for continuing our current research work in the area of high performance microprocessor design.

In late 1994 and early 1995, nearly every major microprocessor vendor announced their “next generation” microprocessor. These companies include Advanced Micro Devices, Digital Equipment Corporation, Hewlett-Packard, Intel, MIPS, Sun, and the Apple/IBM/Motorola PowerPC consortium. Each processor was designed to take each respective company to the “next level” of microprocessor performance. While each company’s current generation offerings may be disparate, the next generation processors are all surprisingly similar, in both implementation and resulting performance. All the processors are superscalar, multiple-instruction dispatch per cycle machines, with varying degrees of out-of-order execution allowed.

This paper will survey the implementations of each of the next generation processors, which are the AMD K5, the DEC Alpha AXP 21164, the HP PA-8000, the Intel P6, the MIPS R10000, the PowerPC 620, and the Sun UltraSPARC. The survey is microarchitectural in nature; there is minimal discussion on the testing, design, or actual fabrication of each processor. The paper first discusses the background of each processor in Section 2.0, to give an idea of each company’s microprocessor design portfolio. From there, the order of evaluation will be similar to an instruction’s flow through the microprocessor. Section 3.0 discusses the instruction predecode and fetch process, which brings instructions into the processor from the cache. Next, an overview of the pipeline implementations of each processor is presented in Section 4.0 to give the reader an overview of instruction execution. Section 5.0 details the complex instruction decode process which is required by the x86-based microprocessors of AMD and Intel. Section 6.0 discusses branch prediction, a crucial element early in an instruction’s execution that determines the next instructions to be fetched. The prediction algorithms, accuracies, compiler support, recovery mechanisms, the number of speculated branches, and the penalties for predicted branches are discussed. Section 7.0 discusses each processor’s method of renaming registers and/or resolving dependencies. Section 8.0 covers instruction dispatch (the process of sending decoded instructions to buffers to await execution), while Section 9.0 discusses

instruction issue, whereby instructions are sent from the buffers to execution units. Section 10.0 details the execution units included in each microprocessor, such as integer/ALU units, floating point units, and load/store units. Several processors even include additional functional units, which are also discussed here. Section 11.0 details cache implementation, including the first level instruction and data caches as well as second level caches. Section 12.0 details each instruction's last stage in the pipeline, whereby they are retired and their results are made visible to architectural state. Section 13.0 discusses each processor's connection to the outside world, via its system bus. Section 14.0 discusses the processors' physical characteristics, from transistor count to power dissipation. Section 15.0 reveals the estimated performance of each processor, and remarks on their relative efficiencies. Finally, Section 16.0 concludes the paper.

2.0 Background

Each of the microprocessors discussed in this paper represent the next-generation iterations of each company's current line of microprocessors. To date, only one processor, the DEC Alpha 21164, is chipping in volume, while the rest are planning volume shipments for the second half of 1995 or the first half of 1996. Each processor was developed to bring the respective company to "the next level" of microprocessor performance. While their goals are similar, each processor's background is somewhat unique.

Two processors, the AMD K5 and Intel P6, are based on the x86 CISC architecture, while the remaining processors all implement their own RISC instruction sets. The PowerPC, MIPS, and SPARC consortiums are "open architectures," permitting multiple vendors to utilize their chips in different systems. HP and DEC, however, are the main consumers of their own processors, using them in proprietary workstations and servers.

2.1 Intel P6

Intel is the undisputed microprocessor market champion, with the largest installed base (hundreds of millions of computers and growing) and the largest number of microprocessors shipped each year. The key to Intel's success has actually been its greatest limitation: its basis on the archaic x86 architecture. Despite its variable length instructions and dearth of architected registers, the x86 platforms have survived due to their enormous installed base of software. By maintaining backward compatibility with each new generation microprocessor, Intel assures itself of staying at the forefront of the x86 market and consequently, on top of the microprocessor sales chart.

The current generation Intel microprocessor is the Pentium, the first superscalar x86 implementation available. Being a two-way superscalar machine, the Pentium was a great leap forward from the single-issue, pipelined 80486 which it replaced. However, even the Pentium's performance was lagging, compared to the RISC processors that were being used in workstations and for scientific applications. Though the x86 CISC architecture has inherent drawbacks, Intel sought to shrink the performance gap between the P6 and the next generation RISC processors. As a result, Intel chose to implement a three-way superscalar processor with what Intel terms "Dynamic Execution." Dynamic Execution refers to the principles all the other next generation processors are including: branch prediction, for speculatively executed instructions, and an out-of-order execution engine. In order to overcome the bane of variable length x86 instructions, Intel decodes each x86 instruction into RISC like operations, which are then fed to a RISC-like core for execution. By incorporating so many of the RISC competitor's techniques, Intel has created an x86-based microprocessor that compares favorably in performance and cost to all the next generation RISC processors.

2.2 AMD K5

Advanced Micro Devices (AMD) has always been in Intel's shadow, trying to chip away a piece of Intel's monopoly of the enormous x86 market. Initially, AMD started out simply cloning Intel processors, and achieved success by selling higher clock-rate versions than their Intel counterparts, but for the same price. However, AMD has always lagged one generation behind Intel, cloning and selling Intel's "last generation" microprocessor while Intel moved on to its next generation.

Always the underdog, AMD decided to take a bold leap forward by not waiting to clone the next generation Intel processor (which at the time was to be the Pentium) and instead developing its own x86-based processor from the ground up. Known inside the company as the "Kryptonite" which was to weaken the strength of "Superman" Intel's market domination, the Krypton-5, or K5 was developed. The K5 combines many of the features used by the next generation RISC processors and uses a more powerful decode logic than the P6 to produce a four-way superscalar x86 machine. Like the P6, the K5 converts x86 instructions on the fly into RISC-like instructions, which execute speculatively and out-of-order in a RISC-like core. While the K5's performance is admirable, it lacks many of the features of next generation microprocessors, such as on-board L2 cache control and system bus management. This was a result of AMD's need to conform to Pentium system designs to permit easy integration into OEM designed systems. As a result of being the market underdog, AMD must follow in the footsteps laid down before it by Intel, and thus suffers some performance penalties. Perhaps

when AMD finally garners enough market share to dictate its own system configurations, AMD will be able to build a processor which is as capable and powerful as the current generation Intel processor in all respects. Still, the K5 is an impressive processor in many microarchitectural aspects, and should not be ignored. The K5's expected sales will probably match that of many of the RISC companies' sales expectations, simply due to the vast x86 market and the relatively small sales volume of RISC microprocessors.

2.3 PowerPC 620

Back in 1991, Apple, IBM, and Motorola formed an alliance in an effort to battle the seemingly unstoppable x86 (read: Intel/Microsoft) juggernaut. The result was the PowerPC alliance, which laid down a roadmap of RISC processors that would cover everything from the low-cost desktop to portable computers to mainstream desktops and high-end workstations and servers. The PowerPC 620 is the last processor planned in the original roadmap, and is intended for the high-end workstation and server markets. The 620 comes right on the heels of the PowerPC 604, a four-way superscalar, 32-bit implementation of the PowerPC architecture (which itself was derived from the IBM POWER architecture), that just began shipping in volume in 1Q95.

The 620 is the first 64-bit implementation of the PowerPC architecture. It does not differ greatly from the 604, whose execution engine it shares. In addition to a 64-bit datapath, the 620 adds larger on-chip caches, an increased number of reservation stations (to permit a larger out-of-order execution window), and on-chip L2 cache and system bus control logic. While its performance may not be as great as the other next generation RISCs, it is more than adequate in comparison to the Intel P6, with which it is to compete. Considering that the 620 is intended to be a high-volume microprocessor, as opposed to those developed by HP, MIPS, Sun, and DEC, the 620 should perform well in its intended market.

2.4 DEC Alpha 21164

Since RISC processors have become "in fashion," so to speak, Digital Equipment Corporation has been at the top of the performance ladder. The DEC 21064 was the first implementation of the Alpha AXP RISC architecture, and was clocked at an (at the time) amazing 150 MHz. The 21064, a 64-bit, two-way superscalar processor, has recently reached 275 MHz in shipping versions. However, DEC knew that it could not survive on the 21064 alone, with other RISC vendors moving on to four-issue machines. Thus came the 21164, a 64-bit, 300 MHz, four-way superscalar implementation of the Alpha architecture. The 21164 was

the first microprocessor to break the 200 SPECint92 performance point, and reaches 330 SPECint92 when running at its highest currently shipping clock rate of 300 MHz. The 21164 is the first “next generation” RISC to actually achieve shipping versions, with the 300 MHz part beginning shipment in March, 1995.

DEC took a totally different approach to obtaining high performance than any of its competitors. Instead of dedicating their efforts to creating complex out-of-order execution engines, Digital chose to optimize the 21164 for clock speed. Wherever possible, tradeoffs were made to reduce complexity to reduce cycle time. As a result, the 21164 forgoes many of the techniques used by the other processors, such as register renaming and out-of-order execution. Their decisions seem to have been for the better at this point in time, however, as their extremely high clock rate has permitted DEC to retain the honor of producing the fastest and most powerful microprocessor in the world. For now, optimizing for speed is winning over optimizing for complexity at the expense of a lower clock rate.

2.5 MIPS R10000

The MIPS R10000 is the next generation in the MIPS open architecture, being the spiritual successor to the R4000. Internally known as the T5, the R10000 was designed to have a large window of instructions available for out-of-order execution. Contrary to its predecessor, the R10000 utilizes shorter pipelines in order to reduce the penalties incurred for branch mispredictions and load-use combinations. Based on the MIPS IV RISC architecture, the R10000 is a full 64-bit processor. Despite having fewer pipeline stages, MIPS expects the R10000 to run initially at 200 MHz, which gives it performance numbers which rival those of the DEC 21164. However, the R10000 is behind schedule, and will be shipping in volume at the earliest in 4Q95. By then, DEC should have a higher speed 21164 shipping, preventing MIPS from taking the microprocessor performance crown away from DEC.

2.6 Sun UltraSPARC

The Sun UltraSPARC is the next generation of the SPARC open architecture, succeeding the SuperSPARC. The 64-bit UltraSPARC is expected to triple the performance of the 60 MHz SuperSPARC, and does so without merely tripling the clock rate (initially expected to be 167 MHz). The design corrects two of the SuperSPARC’s major flaws: the “double-pumped” register file (a register file which must be accessed twice every cycle to accommodate the register windowing feature of the SPARC architecture) and the similarly flawed TLB. Though

the UltraSPARC still executes instructions in order, it has fewer issue restrictions than the 21164, allowing for more instructions to be in flight at once.

Similar to Intel, Sun has the largest installed base of any RISC system vendor. As a result, each new generation processor is guaranteed to be binary compatible with previous generations; doing so assures Sun of a continued large market share in the RISC arena of workstations and servers. To further the appeal of its workstations, the UltraSPARC implements a set of graphics and multimedia instructions on top of the 64-bit SPARC V9 architecture. These instructions permit the processing of video and sound on-chip without the use of external hardware. Doing so permits Sun to sell multimedia systems at lower cost, effectively bolstering its market share in the coming era of multimedia-oriented computing.

2.7 HP PA-8000

The PA-8000 is the next generation Precision Architecture RISC microprocessor from Hewlett-Packard. The PA-8000 is based on PA-RISC 2.0, a 64-bit version of the PA-RISC architecture which debuted in 1986. The PA-8000 succeeds the PA-7200, which was a processor optimized for clock rate, like the Alpha microprocessors. However, the PA-8000 takes a different tack, joining the majority of other RISC vendors in producing a complex out-of-order execution engine at the expense of lower clock rates. To that end, the PA-8000 permits the largest number of instructions to be executed out-of-order at any given time, at 56, compared to the other RISCs. (The closest competitor is the MIPS R10000, which permits 32 instructions to be in flight at once).

One interesting design point of the PA-8000 is its total lack of on-chip cache. HP chose to instead implement large (greater than 1M each) off-chip first level caches, and totally forgo second level caches. This decision was made to improve the PA-8000's performance on business applications, which typically have larger working sets than can adequately be contained in any on-chip cache of a fast microprocessor. As a result, the PA-8000 requires large, extremely fast synchronous memories to buffer the CPU from main memory, but saves silicon on-chip to realize its massive out-of-order execution engine. While performance on benchmarks may lag other processors' due to its longer latency off-chip caches, performance in real-world applications should be superb.

While its performance at 200 MHz is estimated to exceed that of the 300 MHz 21164, the PA-8000 has yet to reach volume shipments. It was the last of the major next generation RISCs to be announced, and is not expected to be shipping in quantity until early 1996. By then, DEC should have higher speed parts which would effectively eliminate the PA-8000's performance edge.

3.0 Predecode and Fetch

Each microprocessor must eventually go off-chip to obtain new instructions, as most programs' working sets are too large to fit in the on-chip cache. For the RISC machines, a known number of instructions are fetched from off-chip every access due to their fixed-length instructions. All the RISC processors discussed here are capable of fetching four instructions worth of bits every cycle from off-chip memory.

The x86 processors, on the other hand, must deal with variable-length instructions, an artifact of the x86 ISA upon which they are based. The K5 pulls in 64 bits per cycle, while the P6 fetches twice that amount, at 128 bits per cycle. Since x86 instructions can range anywhere from one byte to 15 bytes in length, this fetch bandwidth may not even be adequate to deliver one new instruction to the processor per cycle. However, the longer length instructions have been effectively reduced in number via compiler optimizations, so at least one instruction is normally fetched from memory per cycle for the K5, and two or more for the P6.

The variable length instructions can play havoc with the instruction decode process, as the beginning and end of each instruction must be found before even starting the decode. To remedy this situation, the K5 predecodes the instructions as they are fetched from off chip in order to determine the instruction boundaries in the bit stream. In addition, the desired operand sources are found, and the functional unit required for execution is determined for each instruction.

Though the RISC processors do not need to find instruction boundaries, they have also chosen to predecode their instructions as they come on-chip, as this predecode process effectively removes the need for instruction decode. After finding the operand locations and desired functional units, this information is stored, as additional bits, along with the instruction in the instruction (I-) cache. The number of appended bits ranges from four to seven. Thus, every instruction in the cache that is reused is effectively decoded already (for the RISC machines), greatly simplifying the dependency resolution process and speeding instruction dispatch.

AMD K5	21164	PA-8000	Intel P6	R10000	PPC 620	UltraSPARC
5	5	5	n/a	4	7	4

Table 1: Number of predecode bits added

The lone processor which does not predecode instructions is the Intel P6. Intel argues that predecoding instructions adds an unnecessary delay to the fetch process, while the added bits bloat the I-cache. (AMD's additional 5 bits of predecode information per instruction increase the size of the I-cache by 50%). The other processor designers, however, argue that this added delay

is effectively eliminated due to the large number of pending instructions that are available for execution due to their out of order execution engines; the added time for fetching new instructions is minimal compared to the time saved reducing the complexity of parallel instruction dispatch.

From the I-cache, each RISC processor fetches four instructions worth of bits per cycle (including the additional predecode information). Still hampered by variable length instructions, the x86 processors pull in as many bits as possible from the I-cache each cycle, and then decode them into RISC-like, fixed length instructions (see Section 5.0). The K5 fetches a monstrous 208 bits per cycle from the I-cache (128 bits of instructions plus 80 bits of predecode information). Meanwhile, the P6, without any predecode bits, fetches only the 128 bits of instruction data from the I-cache per cycle.

4.0 Pipeline Lengths

As high performance microprocessors, each of these processors is a pipelined machine. The lengths and implementations of pipelines range from the classic 5-stage RISC pipeline to the P6's lengthy, superpipelined 14-stage implementation. Increasing the number of pipeline stages permits the designer to increase the processor's clock rate by reducing the lengths of critical paths; by splitting a long critical path into two or more pipeline stages, a higher overall clock rate can be achieved.

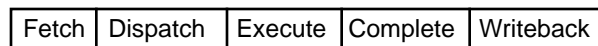


Figure 1: PowerPC 620 pipeline

The PowerPC 620 utilizes the classic 5-stage RISC pipeline, shown in Figure 1. Due to the 620's predecode process, the instruction decode and dispatch can be combined into one pipeline stage. The Execute stage is common to all operations, though they may be of different latencies. However, the use of reservation stations permits the dispatch of one instruction to each functional unit each cycle; if there are no pending instructions in the reservation stations, the instruction can begin execution. The Completion stage is used to complete instruction execution and retire instructions (see Section 12.0). Results of all instructions are available for use at the end of the Completion stage, leading to a one-cycle load-use penalty. Finally, any results not written back to the physical register file during the Completion stage are recorded in the Writeback stage.

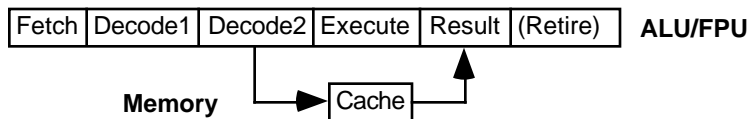


Figure 2: K5 pipeline

The AMD K5 also uses a short, six stage pipeline. The only thing keeping it from using the classic 5-stage RISC pipeline is the x86 ISA to which it must conform; the variable length instructions take two stages to decode, first decoding x86 instructions into ROPs (see Section 5.0) and then dispatching them to the reorder buffer. Cache access occurs in parallel with instruction execution; the K5 generates its address and accesses data in the same cycle. This results in a 0-cycle load-use penalty, which is highly beneficial to an x86-based machine. With a small number of architected registers, most x86 code resorts to numerous memory accesses; by eliminating any load-use penalty, the K5 substantially improves the performance of un-recompiled x86 code.

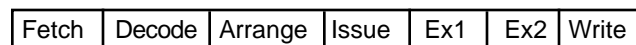


Figure 3: 21164 pipeline

The 21164 pipeline is also relatively short, requiring seven stages. The extra stages, as compared to a 5-stage RISC pipeline, are the Arrange stage and a second Execute stage. In the Arrange stage, instructions are scanned to determine which ones can be dispatched to execution units that cycle. The second Execute stage is mainly for Load/Store instructions to obtain their data from the D-cache. Since a use of the loaded data could be issued the same cycle as the load, there is a one cycle load-use penalty for hits in the L1 data cache.

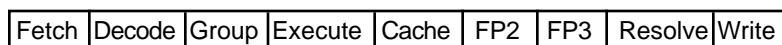


Figure 4: UltraSPARC pipeline

The UltraSPARC uses even more stages, at nine, for all of its instructions. The first five stages are similar to the classic 5-stage RISC pipeline, while the latter stages (FP2, FP3, and Resolve) are used to handle floating point operations. By making all instructions wait the extra three stages before writing back to the register file, FP traps are easier to resolve. With the large number of instructions in flight at any given time, however, these added stages have a minimal impact on performance, while simplifying trap resolution. Since the Cache and Execute stages are still adjacent, a one-cycle load use penalty results.

With their centrally-located instruction queues and reservation stations, the R10000, PA-8000, and P6 all implement variable length pipelines. After the fetch and decode stages, instructions wait for their respective operands and functional units to be available before beginning execution. Since this wait does not affect any subsequent instructions, variable length pipelines can be used to produce results with as little latency as possible. This separation of instruction decode and execution is what is known as a decoupled architecture, which permits execution to start without regard for subsequent instruction's availability. While decoupling is still evident in the PowerPC 620 and K5, it is more apparent in the following three processors, due to their centrally located instruction buffers.

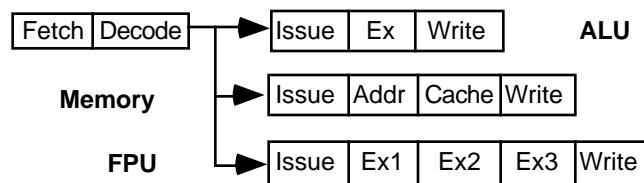


Figure 5: R10000 pipeline

After the requisite Fetch and Decode stages, the R10000's instructions are fed to one of three queues: Memory, Integer, or FPU (see Section 9.0). Instructions in each queue can begin execution at any time, after waiting at least one cycle to be issued to the functional units. Integer operations take two additional stages, thanks to single-cycle ALU operation latencies; FPU operations take four additional stages, due to the three cycle latency of FP operations. Memory accesses are as expected, with address generation and cache access being in separate stages, resulting in a one-cycle load-use penalty.

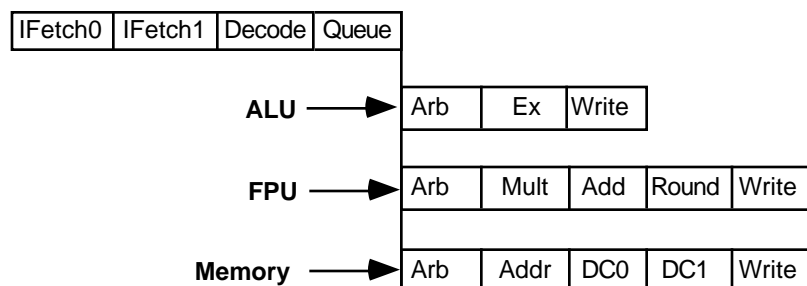


Figure 6: PA-8000 pipeline

Similarly, the PA-8000 implements three different pipelines, one for each of Integer/ALU, FPU, and Memory operations. The PA-8000 requires two cycles to fetch instructions from memory due to the two-cycle latency of its off-chip primary caches (see Section 11.0). Afterwards, instructions are decoded and sent to either the Compute or Memory

queue, where they await the availability of operands and functional units. One cycle is then required to arbitrate for functional unit usage. The Memory pipeline is longer than the R10000's due to the two cycles needed to access the off-chip primary data cache. As a result, all load-use combinations suffer a two cycle penalty. The PA-8000's FPU pipeline is interesting, though, as there is no intermediate rounding stage between FP multiplication and addition. Not only does this save one cycle, it also improves the accuracy of multiply-accumulate instructions.

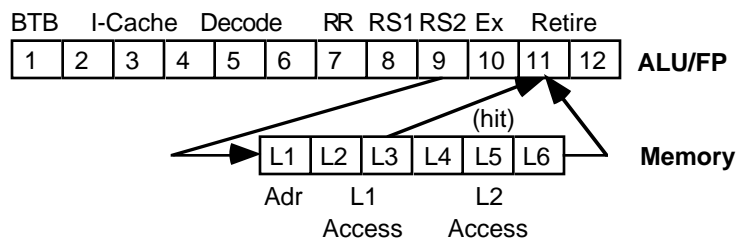


Figure 7: P6 pipeline

Finally, we have the P6's extremely long pipeline. The first stage is for branch target buffer access, to determine if a branch that was fetched should redirect the fetch stream or not. If this is not done as early as possible, the number of instructions that are wasted is potentially very high, due to the two and a half cycles required to actually access the instruction cache. After accessing the I-cache, the next two and a half cycles are used to decode the variable length x86 instructions and convert them into uops (see Section 5.0). Without any predecode information, the P6 requires an extra half cycle to obtain its decoded, RISC-like operations, compared to the K5. Stage 7 is used for register renaming, and in stage 8 the instructions are sent to the global reservation station. Instructions are read from the reservation station in stage 9, and executed in stage 10 if they are ALU or FP instructions. It then takes two cycles to retire instructions, with the first cycle to write the results to the reorder buffer, and the second to update architectural state. Loads and stores execute in a separate pipeline after receiving their operands in stage 9. Stage L1 is used for address calculation, and stages L2 and L3 are used for primary data cache access. Assuming there is a hit in the primary data cache, the instruction can be retired in the next two cycles. If there is an L1 cache miss, the next three cycles are required to access data in the second level cache. Thus, for accesses which hit in the L1 cache, there is a three-cycle load-use penalty.

5.0 Decode

All these microprocessors share a basic superscalar design, with each one dispatching up to four instructions per cycle, except for the Intel P6, which dispatches up to three instructions

per cycle. Before dispatching instructions, the x86 machines first convert the non-uniform length x86 instructions into RISC-like, fixed length instructions, which are called ROPs and uops by AMD and Intel, respectively. The K5 can decode up to four x86 instructions per cycle, whereas the P6 can decode a maximum of three. The K5 has four general purpose ROP converters that translate x86 instructions directly into ROPs. The converters are fed in order by a byte queue, but any instruction in the byte queue can be translated as long as the converters can handle them. Each converter can handle any x86 instruction that requires up to three ROPs for execution. For those x86 instructions requiring more than three ROPs, a microsequencer, MROM, is used to generate the corresponding ROP sequences; the MROM itself can issue up to four ROPs per cycle (though a maximum of four ROPs can be issued at any given time).

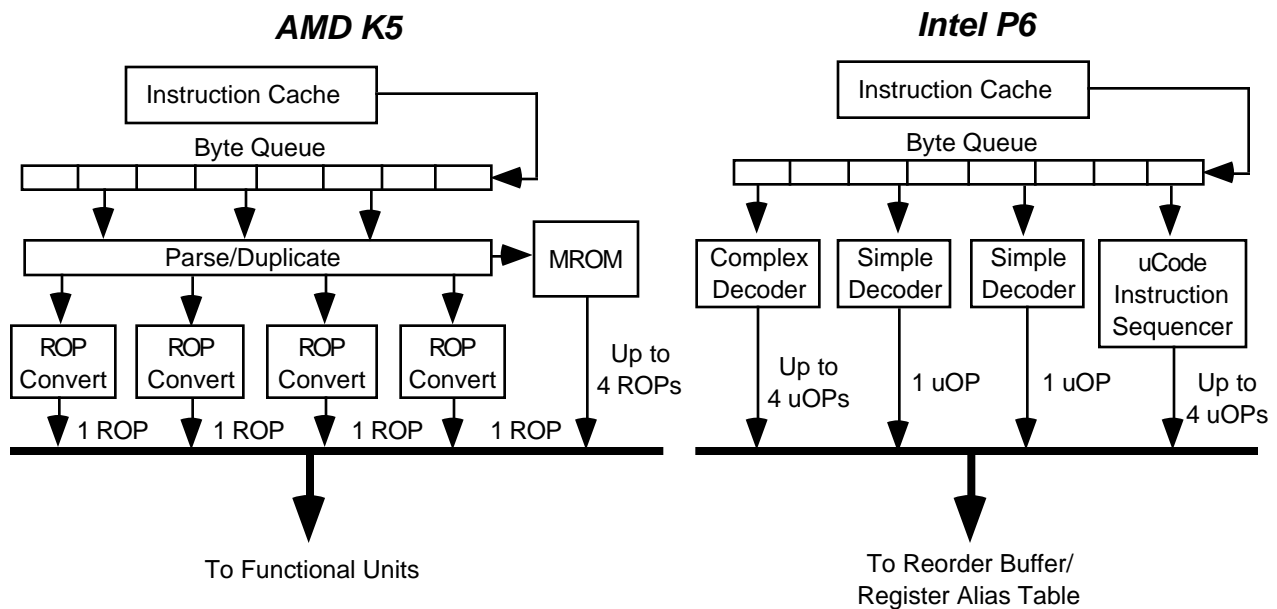


Figure 8: K5 and P6 decode process

Intel's decode process is somewhat more restricted. The P6 has three x86 decoders that are fed in order, where the first one can handle "complex" x86 instructions requiring up to four uops. The other two decoders can only convert x86 instructions which require a single uop to execute. If more than one complex instruction is found in a group of three x86 instructions, the second (and third) complex instruction must wait until it reaches the head of the instruction queue to be translated by the complex translation unit. As in the K5, x86 instructions requiring more than four uops are sent to a microsequencer, which generates the corresponding uops over two or more successive cycles.

The K5's decode process is simpler, as the predecode bits tell the ROP converters the beginning and end of every x86 instruction; thus it is easy to find and decode four ROPs worth of

instructions each cycle. The P6, however, must hope that the x86 instructions are grouped in threes ordered complex-simple-simple. If this is not the case, the P6 will potentially decode only one instruction per cycle. Beyond this decode process, both the K5 and P6 have RISC-like cores, as all instructions are now fixed length.

As discussed in Section 3.0, by using a predecode process, the RISC processors already simple decode process is essentially eliminated. Any required decoding is done in conjunction with instruction dispatch (see Section 7.0).

6.0 Branch Prediction

In order to reduce the penalties imposed by control flow hazards, each processor implements some method of branch prediction. This permits the processor to continue fetching and executing instructions before a branch has been resolved. To redirect the instruction fetch stream as soon as possible (should a predicted-taken branch be encountered), branch prediction is performed during the instruction decode process, as soon as an address containing a branch instruction is encountered. This reduces the number of instructions that are wasted, as the predicted execution path is fetched as soon as possible. The penalty for a mispredicted branch is dependent upon the pipeline stage in which the branch is actually resolved. Should a mispredicted branch be encountered, all speculatively executed instructions and their results must be flushed, returning the processor to a known, non-speculative state for it to resume execution.

6.1 Prediction Algorithm

There are several different prediction algorithms employed by these processors, but the most common is the Smith 2-bit saturating counter [1]. This method keeps track of a branch's execution history using a 2-bit saturating counter, which determines the prediction used for the next encounter of a given branch: strongly not taken, likely not taken, likely taken, and strongly taken. This history is kept in a branch history table (BHT), which is accessed whenever a branch is encountered. The PowerPC 620 and PA-8000 each have a 256 entry BHT, while the R10000 and UltraSPARC use 512 entry BHTs, and the 21164 employs a massive 2048 entry BHT. The larger the BHT, the greater the number of branches that can be tracked and predicted at any given time.

In addition to the BHT, some processors utilize a branch target address cache (BTAC), which contains the predicted taken destination address for each branch that has been encountered. Only predicted taken addresses are cached, as not-taken branches simply fall through and continue fetching from the sequential PC. If a predicted taken branch has a

corresponding entry in the BTAC, the time to redirect the fetch stream is effectively nil, as that address can be moved from the BTAC to the PC in the same cycle that the branch is predicted. The PowerPC 620 and UltraSPARC both utilize 2048 entry BTACs in addition to their BHTs, while the PA-8000 makes do with a 32 entry BTAC.

The K5 uses a simpler prediction algorithm, keeping only a 1-bit history of each encountered branch (taken or not taken for the last execution of said branch). Though the accuracy of predictions is lower with this 1-bit method than the 2-bit Smith algorithm, AMD keeps track of predicted branches by simply appending a bit to each branch located in the cache. Thus, the K5 can effectively keep track of 1024 branch histories, as there are 1024 16B cache lines in its I-cache. AMD argues that the number of predicted branches that are tracked effectively counters their more primitive prediction algorithm; this is more important for business applications which encounter a large number of branches over any given period of time.

The PA-8000's implementation of the Smith algorithm is unique; instead of using a 2-bit saturating counter, as the other processors do, the PA-8000 keeps its branch history in a three bit shift register. The three bits store the last three results of a branch, and the next prediction is based on the majority of those three bits. For instance, if the last three iterations of a branch were taken, not-taken, and taken, the branch would be predicted taken. This implementation is simpler than a 2-bit counter implementation, as all that is required to update branch history is shifting in a bit corresponding to the actual outcome of the branch's execution into the BHT entry for that branch.

The P6 chose to break from common ground and utilize the adaptive branch prediction algorithm proposed by Yeh and Patt [2]. In addition to the 2-bit saturating counter used by the Smith algorithm, a pattern history table is used to help predict more volatile branches. For instance, if a branch had been encountered with a pattern TNTTNNT (where T = taken and N = not taken) previously, and the branch history for the currently examined branch matched this pattern, a more accurate prediction would be made than simply using the 2-bit counter. In this case the 2-bit counter would predict likely not taken, whereas in fact the branch may be known to always be taken after this pattern of branch outcomes. The branch history is held in a 512 entry BHT, and backed by a 512 entry BTAC (which Intel calls a branch target buffer, or BTB).

While the P6 tracks entries in the BTB by the address of the branch instruction, the PowerPC 620 and PA-8000 index the BTAC by the instruction fetch address. Since each processor can fetch up to four instructions at a time, and instructions do not necessarily have to be grouped the same way each time the instruction sequence is encountered, each branch instruction can be in up to four different entries in the BTAC depending upon where it is located within the group of fetched instructions. This helps improve the BTAC's effectiveness, since the position of the branch instruction within a fetch group is affected by the outcome of recent

branch predictions. Thus, this additional recent branch history sometimes provides a useful hint as to whether the current branch is likely to be taken or not.

6.2 Accuracy

The predicted accuracy of each branch prediction algorithm is shown in the table below. The estimated accuracies correspond to each processor's performance on the SPEC92 integer benchmark.

	<i>AMD K5</i>	<i>21164</i>	<i>PA-8000</i>	<i>Intel P6</i>	<i>R10000</i>	<i>PPC 620</i>	<i>UltraSPARC</i>
<i>Algorithm</i>	Last time	Smith	Smith	Yeh	Smith	Smith	Smith
<i>BHT size</i>	1024	2048x2	256x3	512x4	512x2	256x2	512x2
<i>BTAC size</i>	1024	n/a	32	512	n/a	2048	2048
<i>Accuracy</i>	<80%	80%	80%	90%	80%	90%	88%

Table 2: Branch prediction algorithm, implementation, and accuracy

Though four processors utilize the Smith algorithm, their branch prediction accuracies are not equal; this is due to their chosen BHT and BTAC sizes, which affects the number of branch histories that can be tracked, as well as the number of target addresses remembered. Note, however, the high accuracy of the Yeh algorithm despite its smaller BTB (relative to the PowerPC 620, which also obtains a 90% prediction accuracy). This improved performance is due to the Yeh algorithm's adaptive nature; it can more accurately predict erratic branches.

6.3 Compiler Support

The PA-8000 and PowerPC 620 also permit static branch prediction, aided by the compiler. A bit can be set in the instruction which tells whether the branch should be predicted taken or not. If the compiler "knows" more about the future execution of code, it can tell the hardware to override the BHT and instead use the static branch prediction made by the compiler. This static prediction is most beneficial for predicting erratic branches, which the 2-bit counter is more likely to mispredict than not. However, with its adaptive branch history, the Yeh algorithm eschews the need for static prediction, as it most likely will recognize a pattern of branch outcomes as well as a static prediction would, and without compiler support.

6.4 Recovery

When a branch is determined to be mispredicted (after the branch is actually executed), action must be taken to prevent the speculatively executed instructions from modifying architecturally visible state. Most of the processors perform this recovery through whatever mechanism is used to keep track of instructions in flight. In the K5 and P6, this is the reorder buffer; for the PowerPC 620 and UltraSPARC it is the completion unit; the R10000 uses an active list, and the PA-8000 maintains instruction status in its instruction queues. All instructions write back their results to these tracking units, and only when all instructions prior to a given instruction are retired (its results are written back to the register file) can the instruction be retired. In the meantime, the instruction waits in the tracking unit, which contains the result of the operation. If a mispredicted branch is encountered, a flag is simply set on each instruction that follows the branch in program order, nullifying the instruction. When nullified, the instruction does not write to the register file, and thus never changes architecturally visible state. Execution can then resume down the correct path without regard for the speculatively executed instructions.

The K5, PowerPC 620, PA-8000, UltraSPARC, and P6 all use this method of invalidating speculatively executed instructions to resolve mispredicted branches. The fetch stream is then redirected down the correct path, and execution resumes. The R10000 is more innovative in its recovery, thanks to its "back-up" potential. When branches are predicted to be taken, the four instructions in the sequential (not taken) instruction stream are saved off in a "resume cache." This is easy to do since these instructions have already been fetched while the branch instruction was being decoded. Thus, when a mispredicted branch is encountered, its corresponding entries in the resume cache are immediately put into execution, reducing the delay associated with redirecting the fetch stream.

The 21164, with its very limited out-of-order execution potential, resolves branch mispredictions in a more conventional way. Instructions are issued in order and (except for memory accesses) are executed in order as well. Thus, a branch's actual outcome will be discovered before any speculatively executed instructions have a chance to write back to the register file. If a branch is found to be mispredicted, all instructions issued following the mispredicted branch are flushed from the execution pipelines and queues, and execution is resumed from the correct PC.

The R10000 also has a single-entry return-address buffer, which is used to help predict subroutine call returns. The return is of course predicted to be taken, but the address' correctness is increased due to this additional buffer. This process saves a cycle, as the return address need not be read from the register file before being placed in the PC. This method is similar to the

21164's four-entry return stack, which performs the same operations, but permits for multiple, stacked subroutine calls. The PowerPC 620 also has an eight entry Link Register Stack, which is used for subroutine calls and serves the same purpose.

6.5 Number of Speculated Branches

While all the processors have branch prediction mechanisms which permit them to continue execution beyond a speculated branch, several processors have added hardware to permit them to continue execution beyond multiple unresolved branches. This permits the processors to have a larger "window" of instructions from which to extract instruction level parallelism, and thus to keep the numerous functional units busy.

The PowerPC 620, UltraSPARC, and R10000 provide mechanisms for execution beyond four speculated branches. The 620 utilizes four reservation stations in front of its branch unit, permitting four outstanding, unresolved branches to be present at any time. Similarly, the UltraSPARC keeps track of four outstanding branches via four floating-point condition code registers. The R10000 utilizes its resume cache to speculate beyond multiple branches: the resume cache has four entries of four instructions each, holding a total of four sets of four sequential instructions corresponding to each speculated branch.

The PA-8000 tracks all branches in its instruction queues. To aid in recovering from mispredictions, branches are inserted into both the compute and memory queues (see Section 7.0), meaning a total of 28 branches could theoretically be outstanding at any given time. However, the likelihood of 28 branches being dispatched in a row without any computations or memory references in between is near nil; with basic block sizes of five or six instructions, a more reasonable four or five unresolved branches are likely to be in the queues at any given time.

6.6 Penalties

Despite all the efforts to predict the future, every branch prediction algorithm is bound to fail some of the time. The performance hit encountered by a mispredicted branch is dependent upon the amount of time required to find the correct outcome of a branch, added to the time required to restore the machine state and redirect the instruction fetch stream. (In most cases, the time required to restore machine state can be overlapped with the time to redirect the fetch stream). The larger the mispredict penalty, the more desirable it is to have a high prediction accuracy. Due to its extremely long pipelines, the P6's use of the advanced Yeh algorithm is explained, due to its 11 cycle (minimum) mispredicted branch penalty. The other processors have more reasonable mispredict penalties, ranging from two to five cycles.

	AMD K5	21164	PA-8000	Intel P6	R10000	PPC 620	UltraSPARC
<i>mispredict</i>	3	5	5	11	2	2	4
<i>correct</i>	0	1	2	0	1	1	1
<i>hit in BTAC</i>	0	n/a	0	0	n/a	0	n/a

Table 3: Penalties for branch prediction outcomes, in number of cycles

Even branches that are predicted correctly can incur pipeline bubbles, depending upon the stage in the pipeline in which branch prediction occurs. Once a branch is predicted, the earliest the instruction fetch stream can be redirected is in the very next cycle. Thus, most processors must suffer at least a one cycle penalty for correctly predicted branches. Due to their out-of-order execution engines, though, this one cycle is usually covered up performing useful operations.

For processors which utilize a BTAC/BTB, the penalty for branches found in the BTAC is zero cycles, as the target instruction address is readily available. The new PC can be moved from the BTAC to the PC in the same cycle that the branch is predicted, permitting the processor to begin fetching down the predicted speculative path in the next cycle.

The P6, however, accesses its BTB in the first stage of the instruction pipeline, permitting the predicted branch target to be fetched on the next cycle. The K5, on the other hand, includes a pointer to the predicted target in the I-cache along with each predicted branch. Thus, both the P6 and K5 have zero cycle penalties for correctly predicted branches.

7.0 Register Renaming and Dependency Resolution

Being superscalar machines, all these microprocessors must deal with the additional register pressure and data dependencies inherent in issuing multiple instructions each cycle. With up to 56 instructions in flight each cycle, the instructions that have just been fetched and decoded are likely to be dependent upon the results which have yet to be produced. These dependencies can include true (RAW), anti- (WAR) and output (WAW) dependencies. Some of the delay can be eliminated using forwarding (or bypass) paths, but this is only possible for true dependencies that are exactly one cycle apart. In most cases, something more complex must be done to resolve dependencies before instructions can be dispatched.

Most of the processors utilize register renaming to alleviate the register pressure caused by a finite number of registers. The renaming process takes each new definition of a register and assigns it to a new, unused physical register. Subsequent uses of a logical register are mapped onto the physical register by the rename mapping logic.

Though the concept of register renaming is shared amongst all the processors, its implementation is not. The K5 implements register renaming in a reorder buffer. When instructions are dispatched to an execution unit's reservation stations, up to four entries in the reorder buffer are allocated for each of the instructions. Each entry keeps track of the program counter associated with the instruction and keeps a place to hold the result of the instruction, if it produces one. The reorder buffer is implemented as a true FIFO, so entries "farther down" in the buffer are older than entries at the top. Since each instruction has dedicated storage for its result, anti- and output dependencies are eliminated. True dependencies are resolved by searching the reorder buffer for the instruction producing the desired source operand, and waiting for that instruction to complete. As long as no more than 16 instructions are in flight (corresponding to the size of the reorder buffer), there will never be any register pressure.

The HP PA-8000 implements the renaming process in a similar fashion. It contains two 28-entry instruction queues (logically one 56-entry queue) in which all dispatched instructions are stored. Each instruction inhabits a unique entry in the queues, and contains a location to store the result of its execution. Thus, all dependencies are resolved in a fashion akin to AMD's method.

The MIPS R10000 performs register renaming using a mapping table, which maps the 32 logical integer and 32 logical floating point registers onto 64 physical registers each. As instructions are dispatched to the instruction queue (see Section 8.0), the target register is assigned a new physical register from the list of those that are currently unused. The renaming process ensures that only one currently active instruction writes to any given register. When a result is produced, a "busy" bit is cleared for the destination register, indicating that its contents are valid; this unambiguous signal tells all waiting instructions to fetch their now-ready operand. While this process solves anti- and output dependencies, true dependencies are solved by the mapping table; desired source operands are simply mapped according to the mapping table, telling the new instruction which instruction to wait for.

The P6 uses a similar technique for renaming. The Register Alias Table (RAT) determines whether a source operand should be read from the real register file (RRF) or the reorder buffer (ROB). The ROB should be read if an instruction currently in flight (and thus in the ROB) produces the desired operand. Like the K5 and R10000, the ROB contains locations for all instructions in flight (up to 40) to store their results before being written back to the RRF. By waiting for sources from the ROB as opposed to always going to the RRF, the RAT effectively saves one cycle between register definition and usage.

The PowerPC 620 also uses a reorder buffer, located in the completion unit, to track each instruction. An entry in the ROB is allocated for each instruction that is about to be dispatched, and dependency checking is done between instructions in the dispatch queue. The eight GPR

rename buffers and/or the eight floating point rename buffers are searched as operands are fetched from the register file. If an operand is to be written to the ROB by an instruction in flight, the instruction is given the tag of the dependent instruction's rename buffer. If the dependent instruction has just completed execution, but its results have not been committed to the physical register file, the operands are fetched from the dependent instruction's rename buffer. Otherwise, operands are fetched from the register file.

The UltraSPARC also uses its completion unit to handle source operand and bypass control. Operands needed by each instruction considered for dispatch are either bypassed directly from results of previous ALU/Load operations, read from the register file, or bypassed directly from the completion unit. The latter instance occurs when results have not yet been written back to the register file, but are available in the completion unit.

The Alpha 21164 is unique in this group in its lack of register renaming capability. In keeping with its high-clock-rate over complexity argument, the 21164 eschews the added delay due to register renaming by issuing instructions in order and using a large number of bypass paths from the end of execute stages back to the beginning of each execute stage of each FU pipeline. Instructions which have output dependencies are not allowed to issue until the results of the conflicting instruction are produced. Similarly, true dependencies stall the issuing process (discussed in Section 9.0). Since operand reads are always done before operand writes, and the processor executes in order with respect to register file updates, anti-dependencies will never occur. Though there are more issuing restrictions from the lack of register renaming capabilities, the time required to perform register renaming is removed, permitting the high(er) clock rate desired by the 21164 designers.

8.0 Dispatch

The dispatch process takes decoded instructions and places them in buffers, where they wait for operand availability. When the operands are available, the instructions issue to execution units and begin execution. Decoded instructions are dispatched to instruction queues or reservation stations, or are simply not dispatched at all. Instructions that must wait for all dependencies to be satisfied before leaving the decode stage (e.g. in the Alpha 21164) essentially do not have a dispatch process; they issue instructions directly from decode. Thus, the following discussion does not pertain to the 21164 or the UltraSPARC.

All the processors dispatch instructions in order; this is because the decode process consumes the instruction stream in order, and the resulting instructions are dispatched immediately. The rate at which instructions are dispatched is dependent both upon the number of instructions that have been decoded for a given cycle and the maximum number of

instructions that can be buffered in instruction queues, buffers, or reservation stations. If the number of available slots in the instruction buffers is less than the available decoded instructions, one or more instructions must delay dispatch until the next cycle.

The P6 can dispatch the fewest instructions per cycle; while the PowerPC 620, K5, and R10000 can all dispatch four instructions per cycle, the P6 can only dispatch three. This is a limitation due to the P6's ROB, which can only process three uops on each cycle. After passing through the ROB, the instructions (with renamed registers) enter a global, unified 20 entry reservation station. Here, they wait until their source operands are available, at which time they issue to the execution units. The issue process is discussed in Section 9.0.

The HP PA-8000 places up to four decoded instructions in a 56 entry instruction queue. The queue is divided into two separate structures: a compute queue for integer and floating point operations and a memory queue for load/store operations, each with 28 entries. All four entries can be entered into either queue, as long as there are enough empty slots in each. Instructions wait in the queues until their operands are available, at which time they issue to execution units.

The R10000 also uses queue structures to hold instructions awaiting operands. Up to four instructions are dispatched each cycle to one of three queues: integer/ALU, floating-point, and memory (load/store). Each queue contains 16 entries, and can accept all four instructions if there is enough room. Dispatching stalls when one or more of the three queues is full.

After passing through the ROP converters, the K5 dispatches up to 4 ROPs per cycle to the functional units, regardless of operand availability. Every execution unit contains two reservation stations except for the FPU, which has only one, and two load/store units, which each have three. The dispatch process stalls as soon as any ROP is blocked from being dispatched due to the lack of an available reservation station.

The PowerPC 620 also dispatches up to four instructions per cycle to reservation stations, with the same constraints as the K5. However, the 620 has more reservation stations: two for each of the integer and floating point execution units, three for the load/store unit, and four for the branch unit (enabling execution past four speculated branches).

9.0 Issue

The next step in the instruction pipeline is instruction issue to the execution units. This stage is reached after the dispatch stage in which instructions are placed in buffers to wait for operand availability, or directly from the decode unit, as in the 21164 and UltraSPARC. The number of instructions issued per cycle is dependent upon operand availability and the number of available functional units.

In the UltraSPARC, after the decode stage, up to four instructions are sent to a 12 entry FIFO instruction buffer. Since register renaming is implemented through the completion unit, no additional logic is encountered. Up to four instructions are issued from the bottom of the buffer each cycle. As stated before, the instructions issue in order; if an instruction can not be issued due to a register dependency or resource conflict, no subsequent instructions are issued on that cycle. In addition, while the first three instructions can be issued to any function unit, the fourth must be sent to only the branch or floating-point units. Sun argues that not allowing the fourth instruction to issue to every unit greatly reduces the amount of dependency checking required for instruction issue, as well as reducing the number of ports to the integer register file. Thus, to obtain its maximum issue rate of 4 per cycle, the UltraSPARC must issue at least one floating point or one branch instruction every cycle.

The 21164 places decoded instructions (in order) into one of two four-word buffers. Up to four instructions per cycle are issued from the current buffer, with the restriction that two instructions must be integer operations while the other two must be floating point operations. Thus, the 21164 can only achieve 4-way issue when running floating point code. Furthermore, if any instruction in the current buffer can not issue (due to dependencies or FU unavailability), neither it nor any subsequent instructions from that buffer are issued. This preserves the true in-order issue of the machine. Only when all of the instructions in the current buffer have been issued can the instructions in the other buffer start to issue, regardless of whether instructions in the other buffer were ready to execute or not.

The remaining processors are less restricted in their issuing policies. Most can issue as many instructions per cycle as there are available functional units. This is due to their decoupled architectures; with out-of-order issue policies, all dispatched instructions are searched for readiness each cycle, and all ready instructions (limited by available FUs) are then issued to the functional units.

Instructions can be dispatched from any part of the instruction queues in the R10000. Each cycle, instructions are marked executable if their operands have become available. The instructions pass through the register file to pick up their operands and then move on to their respective functional units. Only one instruction can be issued to each functional unit per cycle; though there are five functional units, FP branches are handled without using an FU (as only a bit needs to be checked). Thus, up to six instructions can actually be issued each cycle. If there are two or more executable instructions ready for a particular function unit, the oldest ready instruction is permitted to issue.

The PA-8000 takes a similar approach for instruction issue. Every cycle, up to two instructions are sent to address units and two instructions are sent to computation (integer and FP) units. If there are more than two instructions available of any type, the two oldest

instructions receive priority and are issued to the functional units. After being issued, the instructions must fetch their operands from either other instructions in the instruction queue or from the register file. Simply acquiring operands is a complex process, as several instructions could potentially provide a result for a given register; the most recent update must be found and sent with the requesting instruction to an FU in less than a single clock cycle.

After instructions have obtained their operands, up to five instructions can be issued per cycle in the P6. These instructions can be two calculations, a load address, a store address, and a store data. Only one uop per cycle can be sent to the main arithmetic unit (which contains the FPU and full integer arithmetic unit). The second calculation must be either a simple integer ALU operation (no shifts, multiplies, or divides) or a branch target address calculation. This restriction, similar to the UltraSPARC's issue limitation, reduces complexity with minimal performance penalty. When more uops are ready to execute than the number of available execution units, the oldest ready uops are chosen.

For the K5 and PowerPC 620, which both use reservation stations attached to each functional unit, instructions waiting in reservation stations are issued to begin execution whenever all of their operands are available. Should two or more instructions' operands become available in the same cycle, the oldest pending instruction is given priority to begin execution. The number of possible instructions issued per cycle is five for both these processors, as each has five functional units, each with at least one reservation station attached.

10.0 Functional Units

Each processor has multiple functional units to handle the large number of instructions that are issued each cycle. Most all operations are pipelined, permitting multiple instructions of the same type to be in flight at any given time. The main functional unit types are: integer arithmetic/logic, floating point, and load/store. Several processors even add special-purpose functional units, including dedicated branch units and graphics units.

10.1 Integer Execution Units

The integer units in all these processors perform simple arithmetic and logical operations in one cycle. Depending upon the number of units, less common functions like multiplication, division, and shifting may not be included in every copy. For instance, the K5 has two integer units, where only one has a shifter, and the other has a divider. Similarly, the R10000 contains two integer units, where only one contains a shifter and only the other accepts multiply and

divide instructions. In the ensuing discussion, unless otherwise noted, all functions not mentioned to be unique to a given unit are assumed to be common to both.

The PowerPC 620 has two single cycle integer units which are identical, servicing all integer arithmetic and logic instructions except for multiplies and divides. Multiplication and division are handled in the Multiple Cycle Integer Unit, which eases instruction issue, as two simple integer instructions can be issued together with an integer multiply or divide operation each cycle.

The PA-8000 has two identical integer units, which handle all integer operations (including shifting) except for multiplication. As in previous PA-RISC chips, integer multiplication is performed in the FPUs. The UltraSPARC also follows this model, with two general execution units which can perform all arithmetic, logical, and shift operations. However, the UltraSPARC uses one of its integer units to perform integer multiply and divide, as opposed to resorting to additional FPU usage.

Like the R10000, the 21164 has two integer units, where only one contains a shifter and multiplier unit. The other integer unit, though, is the only one which can handle branches. Thus, while the 21164 can pair most integer operations for issue, it cannot group two shifts, two multiplies, or two branches in the same cycle.

The P6 takes a slightly different approach to integer FU grouping. The P6 has a main integer unit, which is fed by the same issue bus as the FPU; this integer unit contains an ALU, shifter, multiplier, and divider. The second integer unit performs only simple calculations (no shifts, multiplies, or divides) or branch target address calculations.

10.2 Floating Point Execution Units

All of these processors have at least one pipelined FPU which handles floating point addition and multiplication. All the FPUs are fully IEEE 754-1985 compliant for both single- and double precision operations. Most often, long latency operations like divide and square root (which may not even be supported in hardware at all) are non-pipelined, but execute in parallel with the pipelined addition and multiplication instructions.

Both x86 processors contain only one floating point unit, due to the x86 market's concentration on integer as opposed to scientific code. In order to become more competitive in the server market, however, Intel has imbued the P6 with an improved FPU, compared to the Pentium, that executes FP adds in three cycles and FP multiplies in five. Adds and multiplies are pipelined, and can be executed in parallel with long-latency operations such as FP divide and square root. The K5, on the other hand, chose to save silicon and instead optimize its integer performance. Its FPU has a longer latency than the P6, taking seven cycles for FP multiplies.

The other latencies for floating point operations in the K5, however, were not available at the time of this report.

The RISC processors, however, have always been strong in floating point/scientific code performance, and this continues in the new generation of processors. The PowerPC 620 has a single double precision FPU. All FP operations are performed in double precision, resulting in no speed improvement for single-precision calculations. All operations are pipelined with a three cycle latency for adds and multiplies. FP divides take 18 cycles to complete, and a hardware square root function takes 22 cycles. FP loads have been reduced in latency as compared to the PowerPC 604, from three cycles to two.

The R10000 is more aggressive in devoting hardware to FP operations, including two FPUs: one for addition and similar functions, and another for multiplies and long latency operations such as divide and square root (which are not pipelined). The latter unit can execute a multiply, divide, and square root in parallel at once, but only one operation can write back its results each cycle, due to its single port to the register file. The FP add and multiply have latencies of only two cycles, helping boost the R10000's SPECfp92 rating. FP loads have a latency of three cycles. Finally, the MIPS IV instruction set includes a multiply-add instruction, which the R10000 executes with a total latency of four cycles; the result of the multiply is fed directly to the add pipeline before a result is returned.

The 21164 also has two FPUs, one to handle FP addition and division, and one to handle FP multiplication. Though it does not directly implement a multiply-add instruction, the 21164 is capable of issuing both an add and a multiply in the same cycle, reducing the potential penalty of not implementing this instruction in the ISA. Addition and multiplication take four cycles, improving upon the 21064's 6 cycle latencies. FP divides are not pipelined, and have latencies of 10 cycles for single precision operations and 23 cycles for double-precision operations. There is no hardware square root function, so it must be emulated in software. There are, however, two floating point load data pipelines and one FP store data pipeline, allowing FP loads and stores to execute in parallel with floating point operations.

	AMD K5	21164	PA-8000	Intel P6	R10000	PPC 620	UltraSPARC
Add (SP/DP)	n/a	4	3	3	2	3	3
Mul (SP/DP)	7	4	3	5	2	3	3
Div (SP/DP)	n/a	10/23	17/31	18	11/18	18/22	12/22
Sqrt (SP/DP)	n/a	n/a	17/31	29	17/32	18/22	12/22

Table 4: Latencies for floating point operations, in cycles

The UltraSPARC goes off the deep end in implementing a total of five FP units: one each for FP addition, multiplication, and division/square root, and two for special graphics functions

(discussed in Section 10.4). Like the PowerPC 620, all FP operations complete in three pipelined cycles, except for divide and square root, which each take 12 cycles for single-precision calculations and 22 cycles for double precision.

In keeping with its two-of-everything philosophy, the PA-8000 has two complete floating point units, which each execute a multiply-accumulate instruction in just three cycles. This savings of one cycle over the R10000 implementation is due to the lack of an intermediate rounding step between the multiply result and the addition. An added benefit of this savings is the improved accuracy of the operation. Single- and double-precision multiplication and addition have latencies of three cycles. Since each FPU can perform both addition and multiplication, two FP multiply-add instructions can be executed in parallel. Furthermore, both FPUs contain divide/square root units, which can execute in parallel with multiply-accumulate instructions. Divides and square roots are not pipelined, and have latencies of 17 cycles for single-precision operations and 31 cycles for double-precision operations.

10.3 Load/Store Units

Due to their high clock rates, the efficient execution of loads and stores is critical to reducing the bottleneck of going to memory, be it cache or main memory. To this end, each processor implements complex, dedicated load/store units. For instance, to handle the numerous addressing modes defined by the x86 instruction set, the K5 contains dual load/store units, each capable of supporting all complex x86 addressing modes. The units perform full x86 address calculations in addition to data cache access, all in one cycle. As a result, for data cache hits, there is no load-use penalty; data loaded by one instruction can be used by the next without stalling.

The PowerPC 620 has a single load/store unit, but allows loads and stores to complete out of order. If a store address matches the address of a speculatively executed load, the instruction pipeline is flushed, and the load is re-executed. No loads or stores that access non-cacheable (I/O) segments are executed speculatively. Loads are sent to a five-entry pending load queue that provides load/store address collision detection. Speculatively executed stores reside in a five-entry finished store queue until they are verified to not collide with any pending loads. From there, the stores enter a six-entry completed store queue, where they reside until the store instruction is retired in program order; the store result is then written out to memory.

The R10000 uses a similar approach for its load/store unit, with its use of an address queue. The queue contains 16 entries organized as a circular FIFO. Instructions can be issued to memory in any order, but must be written to or removed from the queue in sequential order. Up to four instructions can be written to the queue every cycle. The FIFO maintains the program's

original memory reference sequence, easing the computation of address dependencies. Should a memory dependency, cache miss, or resource conflict occur, the address queue re-issues the stunted instruction, along with all younger instructions, until it is completed.

The load/store unit is the only place where the 21164 allows limited out-of-order execution, by allowing instructions to continue executing while cache misses are being serviced. Like the PowerPC 620 and R10000, the 21164 utilizes a six-entry miss address file (MAF) that captures information on loads that miss in the data cache. The MAF acts as a FIFO buffer containing previously issued loads that have missed in the data cache. If a subsequent load is pending data from the same cache line as a stalled load in the MAF, the two loads are merged and processed at the same time when the cache line returns. By merging loads, the six-entry MAF can keep as many as 21 loads pending at once. (Each MAF entry can contain four loads except the last entry, which can hold only one). The MAF entry allows data returned from memory to be written directly to the register file as well as to an execution for any instruction waiting for the load data. The MAF also contains a six-entry write buffer for stores that miss in the data cache. As for loads, two stores to the same cache line are merged, improving performance for programs that generate writes to sequential addresses. For load instructions that issue a cycle after a store instruction to the same address, the load instruction must be re-issued to prevent the load from obtaining old data; there is no store-forwarding available via the MAF.

The UltraSPARC also uses queue structures to negotiate its memory accesses. Up to nine loads and eight stores can be queued at any time. Loads can always bypass stores unless there is an address dependency, while stores are always executed in order. As in the 21164, this is the only place out-of-order execution is allowed in the UltraSPARC. Loads can also bypass loads in the queue, due to the non-blocking data cache. A rate of one load or store from/to the external cache can be sustained per cycle. The load buffer is implemented as a circular queue. The eight-entry store buffer performs store compression when the last two entries in the buffer are to the same 16 byte block in memory.

The PA-8000 has dual address generation units, which output their addresses to an address reorder buffer (ARB). The ARB is 28 slots deep, with each slot associated with a slot in the instruction queue. Once the address is in the ARB, if no other instruction is arbitrating for access to the appropriate bank in the data cache (see Section 11.2), the access is launched immediately. Thus, loads can execute speculatively. In the event a load misses in the cache, it begins arbitrating for access; arbitration is granted based on the age of the originating instruction, not the length of time a load has been in the ARB. This permits younger instructions to continue execution, while older queued instructions are returned to execution as soon as possible. The store queue can hold up to eleven doublewords of write data for each bank of the data cache. Writes to the data cache are performed during idle cycles or when other stores are performing

cache lookups, reducing cache contention. Stores to sequential addresses of less than doubleword size may be merged into a single cache line. Younger loads may bypass data directly from the store queue. When a speculatively executed load is found that reads from a potential store address, the load and all younger instructions are flushed from the instruction queue and re-executed. When a load calculates its effective address, all older stores compare their addresses against the load address, and if they match, the load waits until the store data is available. Like the PowerPC 620, store data is written to the cache only when the store instruction is retired.

The P6 has two address generation units, one for loads and one for stores, and generates addresses for all possible modes in one cycle. Store data is placed in a separate store data unit, while generated addresses are placed in a memory reorder buffer (MOB), where they await the opportunity to access the cache. If the MOB is empty, a load will go directly to the data cache, but will take an additional three cycles (assuming a cache hit) before the loaded data is available for use. If the access at the front of the MOB misses, subsequent accesses will continue to execute, thanks to the P6's non-blocking caches (see Section 10.2). Loads can arbitrarily pass loads, but stores are always executed in order. Loads can only pass stores if the two addresses are verified to be different. The P6 can only handle one load per cycle, contrary to the other processors, which can handle two or more; this was a design decision of the P6 developers, as the entire processor is designed for one load per cycle. However, due to the MOB, it is more likely to be able to produce one memory result per cycle than schemes like that used in the K5, which do not permit speculative loads to execute.

10.4 Other Functional Units

For most processors, integer, floating point, and load/store functional units are sufficient. The AMD K5, PowerPC 620, and UltraSPARC designers, however, felt differently. The K5 and PowerPC 620 each implement a separate branch processing unit (BPU), which handles branch resolution and target address calculations independently of the integer execution units. With four reservation stations feeding the 620's BPU, execution can proceed in the rest of the processor through up to four speculated branches. Whether the K5's BPU had reservation stations attached or not was not available at the time of this report.

Viewing multimedia as a key computer market of the future, the UltraSPARC includes two graphics functional units, one for addition and one for multiplication. While other processors, such as the R10000, implement special graphics-oriented instructions, the UltraSPARC provides the execution units to handle these added instructions. The additional SPARC instructions operate on 8-, 16-, and 32-bit data types in parallel; for example, up to eight

8-bit results can be calculated by a single instruction per cycle. With the additional functional units, Sun expects the UltraSPARC to decode MPEG-2 video at 30-frames per second and perform H.320 video encoding at a level adequate for desktop videoconferencing, all without additional hardware.

11.0 Caches

Caches are crucial to the high performance of these processors, as they act as high speed buffers between the CPU and slower main memory. Most processors have at least one level of cache on chip, with caches dedicated to both instructions and data. To achieve high performance, several processors have even implemented two levels of the cache hierarchy on-chip. For those that make do with one level of on-chip cache, all except the K5 include dedicated logic to control an off-chip second level cache. The PA-8000, however, is unique, as it has no on-chip cache whatsoever. HP's reason for using this design and its resulting implementation of large, first level off-chip caches will be explained in the Sections below.

11.1 Instruction Cache

The instruction cache stores program instructions fetched from main memory. Depending upon the size of the cache and the cache's implementation, larger and larger working sets can be accommodated at any given time, reducing the need to go to main memory. This permits the processor to run at the speed of its instruction cache (I-cache), as opposed to the speed of main memory.

	<i>AMD K5</i>	<i>21164</i>	<i>PA-8000</i>	<i>Intel P6</i>	<i>R10000</i>	<i>PPC 620</i>	<i>UltraSPARC</i>
<i>Size</i>	16K	8K	1M - 4M	8K	32K	32K	16K
<i>Associativity</i>	4-way	direct	direct	4-way	2 way	8-way	2-way
<i>Line Size</i>	16B	32B	64B	32B	64B	64B	32B
<i># bits/cycle</i>	208	148	148	128	144	156	144

Table 5: Instruction cache characteristics

The 21164 utilizes a small, 8K direct mapped I-cache, with a 32B line size. The DEC designers could not utilize a cache larger than 8K with the current CMOS process, however, as they required a fast, single-cycle I-cache to feed instructions to the execution engine every cycle. The instruction translation lookaside buffer (TLB) is single ported and contains 48 entries. Four instructions worth of bits (148) are fetched from the I-cache each cycle.

The P6 also has an 8K I-cache with a 64B line size. However, it is implemented as a 4-way set associative, virtually indexed and physically tagged cache. To translate addresses for cache accesses, the P6 contains a 32 entry I-TLB, which is fully associative and single ported. 128 bits are read from the I-cache each cycle.

The K5 uses a 16K, 4-way set associative instruction cache, with a line size of 16B. The cache is actually 24K in size, but AMD calls it an "effective" 16K cache. The added size of the cache is due to the predecode bits which the K5 stores with each instruction in the I-cache. This is the cache bloating which Intel referred to, and thus avoided by not implementing a predecode process in the P6. The I-cache is virtually addressed and virtually tagged, requiring no translation before cache accesses. 208 bits (128 instruction bits plus 80 bits of predecode information) are read from the I-cache each cycle.

The UltraSPARC also utilizes a 16K I-cache, with a line size of 16B. The cache is pseudo-two-way set associative, utilizing a simple LRU replacement algorithm. A set prediction mechanism is used to access the I-cache, which simplifies the design and makes the access time the same as a direct mapped cache. Only one set of the cache is fetched each cycle, saving power and cache cycle time; a two cycle penalty occurs for a set misprediction. Address translation is handled by a 64-entry TLB and a single-entry micro-TLB. Four instructions (144 bits) are read from the I-cache each cycle.

The R10000 chose to use larger caches than its predecessor, doubling the size of the I-cache from 16K to 32K. The I-cache is 2-way set associative with a 64B line size. Four instructions worth of bits (144) are fetched from the I-cache each cycle. A unified TLB is shared with the data cache (D-cache), and contains 64 pairs of entries, which map pairs of adjacent pages. In addition, there is a micro-TLB which contains eight single-page entries for use by the I-cache only.

The PowerPC 620 also uses a 32K I-cache with a 64B line size and effective 8-way set associativity. The 8-way associativity is not implemented in the traditional method utilizing muxes; instead, a content addressable memory (CAM) is placed in front of each of the eight sets of the cache, and only the set containing the desired instruction data fires at any given time. This procedure reduces the power consumption for the large cache as well as eliminating the wide busses and multiplexers of a traditional design. Simulations have shown that this design has about the same hit rate as a traditional 8-way cache, although performance varies depending upon the data set. Dual 64-entry TLBs (which act as a unified 128 entry TLB) translate effective addresses into physical addresses for both the I- and D-caches. Four instructions (156 bits) are read from the I-cache each cycle.

As mentioned earlier, the PA-8000 strays from the pack by utilizing no on-chip caches. Instead, HP chose to utilize large, fast, off-chip caches. The I- and D-caches are a minimum of

1M in size, and must be implemented using high speed, synchronous SRAMs with registers on its inputs and outputs. Though the latency is two cycles, through the use of pipelining, the caches still support one access per cycle. To further improve the speed, the caches are direct-mapped. HP argues that the use of a larger off-chip primary cache is beneficial to real-world applications, whose working sets are often larger than any cost-effective on-chip cache could contain. Thus, the space saved on-chip from the lack of cache was dedicated to the massive 56-entry instruction buffer; this large window of potential out-of-order execution effectively reduces the latency required to fetch new instructions, as some instructions are likely to be executable while waiting for the off-chip cache accesses.

11.2 Data Cache

Data caches are even more crucial to processor performance than instruction caches, as instructions can stall while waiting for earlier instructions to obtain data from memory. While many processors utilize the same implementation for I- and D-caches, several processors imbue their D-caches with more features, increasing efficiency and hit rate.

	<i>AMD K5</i>	<i>21164</i>	<i>PA-8000</i>	<i>Intel P6</i>	<i>R10000</i>	<i>PPC 620</i>	<i>UltraSPARC</i>
<i>Size</i>	8K	8K	1M - 4M	8K	32K	32K	16K
<i>Associativity</i>	4-way	direct	direct	2-way	2-way	8-way	direct
<i>Line Size</i>	16 byte	32B	64B	32B	32B	64B	32B
<i>Interleaving</i>	4-way	n/a	2-way	4-way	2-way	2-way	n/a
<i>Non-blocking</i>	n	y	y	y	y	y	y

Table 6: Data cache characteristics

The K5 uses an 8K, 4-way set associative data cache with 16B lines. It supports two accesses per cycle, thanks to its interleaved design. The D-cache is interleaved four ways; as long as two accesses are not to the same way (bank) in the D-cache, they can be executed in parallel. Since there are two access ports, up to two loads or stores or a combination of the two can be executed every cycle. Two accesses to the same bank are allowed if they are to the same cache line. Like its I-cache, the K5's D-cache is virtually addressed and virtually tagged, requiring no translation before cache access. However, a single set of physical tags is shared between both the I- and D-cache, and are used for bus snooping in multiprocessor environments.

The 21164 also utilizes an 8K direct-mapped data cache, with a 32B line size. The D-cache is dual-ported and non-blocking, allowing two references per cycle. The data TLB is similarly dual ported, containing 64 entries. Similarly, the P6 uses an 8K D-cache with a 32B line size. The D-cache is two-way set associative, virtually indexed, and physically tagged. It is

non-blocking, supporting subsequent accesses while a pending access is waiting for a cache reload. Like the K5, the D-cache is interleaved four ways, and supports one load and one store per cycle as long as they are to different banks. Since the P6 is built for one load per cycle, only one read port for the D-cache is necessary. Address translation via the 64 entry, dual ported, fully associative TLB occurs in parallel with the cache access.

The R10000 and UltraSPARC also implement non-blocking D-caches, both with 32B line sizes and cache sizes 32K and 16K, respectively. The R10000 uses a two-way set associative mapping, with a 32B line size, while the UltraSPARC chooses to remain direct-mapped for improved speed. As stated earlier, the 64x2 entry unified TLB in the R10000 is shared with the I-cache, while the UltraSPARC uses a dedicated 64-entry TLB for data cache accesses. Both caches are single ported, and can support one load or store per cycle.

Like its I-cache, the PowerPC 620's D-cache is 32K in size, 8-way effective associative with a 64B line size. The 8-way associativity is again implemented using CAMs, making for an "effective" direct mapped cache. It is nonblocking and, like the K5 and P6, interleaved, though only 2 ways. Accesses to separate banks can proceed in parallel each cycle, yielding up to two load or store results every cycle.

11.3 Level 2 (L2) Cache

The level 2 (L2) cache is nearly as critical as the L1 caches, as it provides another level of buffering between the CPU and slower memory subsystem. Processors which have small L1 caches, like the P6 and 21164, benefit from larger, fast L2 caches. As a result, Intel and Digital have chosen to implement their L2 caches either in the same die cavity (as part of a multi-chip module (MCM)) as the CPU or as part of the CPU die itself, respectively. Though their intents were the same (high processor performance), their reasons for providing on-chip L2 caches were different. Intel chose to combine its unified, 256K 4-way set associative L2 cache with the P6 core to ease OEM system designer's inclusion of the P6 in new systems. Often, it is difficult to design an interface to a fast L2 cache; it is also difficult to gauge the correct size for the L2 to obtain optimum performance at a given price point. Intel has thus removed this burden by including the L2 on the same chip, saving the OEMs desperately needed R&D money and maximizing their already slim profit margins.

Digital, on the other hand, could care less about OEMs, since they are the primary consumer of their own chips. DEC needed a large, fast L2 cache on-chip to compensate for its small, 8K L1 caches. The L1 caches had to be small enough to permit single-cycle accesses to feed its tremendously fast computation engine. But, due to their small size and direct-mapped implementation, many cache misses were likely to result. Thus, a fast L2 cache was needed. By

building the L2 on-chip, it can run at the speed of the processor core with minimal latency. Implemented as a unified cache (for both instructions and data), the 96K three-way set associative cache offers higher hit rates than split caches of the same size. By implementing the two-level memory hierarchy on-chip, the 21164 avoids the need for a large dual-ported off chip memory, and in fact reduces the need for any external cache at all. In this way, system cost is reduced, as with the P6, with a performance reduction of less than 10% for many applications.

	<i>AMD K5</i>	<i>21164</i>	<i>PA-8000</i>	<i>Intel P6</i>	<i>R10000</i>	<i>PPC 620</i>	<i>UltraSPARC</i>
<i>Size</i>	n/a	96K	n/a	256K	512K-16M	1M-128M	512K-4M
<i>Associativity</i>	n/a	3-way	n/a	direct	2-way	direct	direct
<i>Line Size</i>	n/a	64B	n/a	32B	64B	64B	64B
<i>On Chip</i>	no	yes	no	yes	no	no	no
<i>Bus width</i>	n/a	128 bits	n/a	64 bits	128 bits	128 bits	128 bits
<i>Separate bus</i>	no	yes	n/a	yes	yes	yes	yes

Table 7: L2 cache characteristics

The L2 caches for the R10000, PowerPC 620, and UltraSPARC are all very similar, with 64B line sizes and a minimum size of 512K (1M for the PPC 620). To keep access speed high, the caches are either direct mapped or two-way set associative. Each of these processors have on-chip L2 cache control, along with a dedicated 128-bit bus from the processor to L2 cache. This permits cache accesses to occur in parallel with other system I/O accesses. In addition, these accesses can occur at the speed of the processor, as this is a single, dedicated bus; the system bus must be shared with many peripherals, most of which would not benefit from (or even be able to use) a system bus running at the processor's internal speed of 133 MHz or more. Thus, though the L2 caches are not on-chip, their interface to the processor is nearly as good, and permits the size of the L2 to be varied depending upon the intended applications for the system.

Neither the K5 nor the PA-8000 provide control or a dedicated L2 bus on-chip. The K5, which was designed to be compatible with the current generation Pentium's system bus, relies on off-chip logic to control the cache interface. Furthermore, the K5's system bus is shared with the cache, reducing the efficiency of the L2 cache interface. This was the bane of trying to remain compatible with Intel to ease the K5's inclusion in OEM systems.

The PA-8000, on the other hand, does not even need L2 caches, due to its massive (greater than 1M) L1 caches. The PA-8000 misses in its L1 cache at most as often as its competitors miss in their L2 caches; thus, adding another level of cache is unnecessary, and would greatly increase system cost before any benefit would be seen. Due to the L1 cache's extremely low latency of two cycles, while the PA-8000 must always take a two-cycle hit for every access to its cache, its competitors (not counting the P6 and 21164) must wait at least six

cycles to access its L2 cache on an L1 miss. HP felt that this was a reasonable trade off, especially considering its benefits when dealing with non-scientific, business-oriented applications.

12.0 Retirement

The last stage in an instruction's execution is retirement. This is when an instruction's results are written back to the architected registers and architectural state is updated. Once an instruction is retired, its results are guaranteed to be correct. With their out-of-order execution engines, however, these processors can not simply retire instructions as they complete execution. They must order instruction retirement so that the instructions retire in program order, preserving the correctness of the program assumed by programmers.

The same basic method of retirement is used by all these processors. Instructions which have been sent to execution units are tracked according to their location in program order. When instructions complete, they wait in buffers until they are retired. An instruction is retired only when all instructions preceding it in program order have been successfully retired. This guarantees the in-order "execution" of instructions which a programmer expects. Should any faults occur (branch mispredictions, exceptions, interrupts, etc.) the faulting instruction is not retired and instead is flushed from the buffer. If the fault was due to a branch misprediction, all instructions younger than the faulting instruction are flushed, as they were executed down the incorrect branch path. If the fault was due to an exception or interrupt, the instruction is simply re-issued, and when it completes successfully, can then be retired. By permitting the instructions to execute out of order but enforcing their in-order completion, as many instructions as possible are permitted to be in flight at any given time. This fact is a critical for every data-flow driven processor to obtain high performance.

The K5 has a 16-entry reorder buffer (ROB), which keeps track of executing instructions and their results, performing register renaming as needed to avoid output dependencies. Since each entry holds an instruction and space for its result, up to 16 instructions can be in flight at any one time. Results are not written to the register file unless they are guaranteed to be correct; instructions who require operands from instructions contained in the ROB obtain them from the ROB directly. Up to four instructions can be retired each cycle.

	AMD K5	21164	PA-8000	Intel P6	R10000	PPC 620	UltraSPARC
Retire Width	4	n/a	4	3	4	4	4
#Inst in flight	16	21 ld/6 st	56	40	32	16	9 ld/8 st

Table 8: Retirement characteristics

The PowerPC 620 also has a 16-entry ROB, which it places in what is called a completion unit; up to 16 instructions can be in flight at any given time. In addition to the register usage tracking implemented in the K5's reorder buffer, the 620's reorder buffer also keeps track of the low-order 12 bits of the effective address for loads and stores. If a store address matches the address of a load that was executed out of order, the instruction pipeline is flushed and the load is re-executed. Up to four instructions can be retired each cycle.

The R10000 uses an active list to track instructions which have been executed. An instruction is retired when all preceding instructions have been retired; up to four instructions can be retired per cycle, with a maximum of one store. To retire an instruction, the processor frees the old copy of the destination register and writes it out to the register file, allowing it to be reused. If an exception is encountered, the active list is used to undo the results of all younger instructions that have completed out of order. Instructions can be undone at a rate of four per cycle, for a maximum exception latency of eight cycles. To reduce the recovery time after mispredicted branches, the processor saves the register mapping table and other important state in shadow registers whenever a branch is speculatively issued. This permits the processor to resume execution immediately after a branch misprediction is determined, without repairing the active list entries four at a time. Four sets of shadow registers are provided to support the four speculated conditional branches that can be in flight. The active list contains 32 entries, permitting 32 instructions to be in flight at once.

The P6 contains a 40 entry ROB, permitting 40 instructions to be in flight, but is only capable of retiring three instructions each cycle. Since up to five instructions can complete execution every cycle, many instructions can wait in the ROB several cycles after completion to be retired. The retirement process takes two cycles: the first cycle is used to read the instruction pool to find the potential candidates for retirement and determine which of these candidates are next in the original program order; the results are then written to the real register file in the next cycle.

The PA-8000 tracks instruction execution via its 56-entry instruction queue. Every cycle the status of instructions in the queues is updated. The instructions are then scanned, checking for completion. Up to four instructions, in program order, can be retired per cycle. If an exception or mispredicted branch occurs all subsequent instructions in the instruction queue can be invalidated in a single cycle.

The UltraSPARC and 21164 both issue instructions directly from the decode stage, and only after their operands are available. As a result, execution proceeds in order, except for memory accesses. The UltraSPARC can queue up to eight stores and nine loads at any given time, while the 21164 can have 21 pending loads and six writes at once. Otherwise, instructions are retired in program order in a fashion similar to the other processors. With its dedicated

completion unit, the UltraSPARC can retire up to four instructions each cycle; the 21164 simply writes back results to the register file whenever instructions complete, as they are executed in order (except for memory references).

13.0 System Interface

While internal processor optimizations are important, a processor's connection to the outside world is just as critical to overall system performance. As a result, each processor, save the AMD K5, has implemented on-board bus control logic. Doing so permits bus arbitration to be processed at the speed of the CPU instead of the bus itself, which in most cases is much slower than the CPU's internal speed.

	<i>AMD K5</i>	<i>21164</i>	<i>PA-8000</i>	<i>Intel P6</i>	<i>R10000</i>	<i>PPC 620</i>	<i>UltraSPARC</i>
<i>MP ready</i>	n	y	y	y	y	y	y
<i># processors</i>	n/a	4	8	4	4	4	4
<i>Protocol</i>	n/a	split-trans	split-trans	split-trans	split-trans	split-trans	split-trans
<i>Width (bits)</i>	n/a	128	64 multiplex	64	64 multiplex	128	128
<i>Throughput</i>	n/a	1.6 GB/s	960 MB/s	528 MB/s	1.6 GB/s	1.1 GB/s	1.3 GB/s

Table 9: System bus characteristics

All these processors, except the K5, support split-transaction bus protocols, which permit multiple transactions to occur on the bus at one time. A processor which makes a request on the bus can simply leave the bus and continue processing until the request returns with the desired data. As a result, there is a minimal performance hit to access the system bus.

Each bus is either 64 or 128 bits wide. This corresponds to the amount of data that can be transmitted on the bus at once. Some busses, like the PA-8000's Runway Bus and the R10000's Avalanche Bus, are 64-bit multiplexed busses, where 64 bits of address bit are sent in one cycle and 64 bits of data bits are sent on other cycles. This is in contrast to the 128 bit data-only busses, which require additional busses for address bits. However, bus throughput is a function of both clock speed and bus width; as seen in Table 9, above, the processors with high clock rates and 128 bit busses achieve the highest bus performance. (See Section 14.0 for clock speed information). Also note that these are system bus throughputs under ideal conditions at the highest possible clocking rates; high amounts of communication combined with lower clock rates will substantially decrease average throughput.

Another important consideration for system bus design was the incorporation of multiprocessor support on each processor. This permits a number of other processors (typically 4) to be attached directly to the system bus to form a high performance multiprocessor system.

Thus, it was critical to obtain the high throughput for these busses to facilitate communication between various processors. All the processors (save the K5, again) implement a MOESI cache coherency protocol on-chip [3] or the MESI subset, to handle communications between the different processors.

The K5 lacks dedicated bus control logic on-chip, as it was designed to be compatible with the P5 (as discussed in Section 11.3). In fact, the K5 is pin compatible with the P54C version of the Pentium, making the incorporation of the K5 into OEM systems as simple as a chip swap. On the down side, external logic must be used to control the bus and to add multiple processor support to the system. Furthermore, the external logic runs at the speed of the bus, which is typically 1/2 of the CPU's core speed or less, greatly reducing throughput.

14.0 Physical Characteristics

The physical implementation of each of these processors is very similar. All use 0.5 micron CMOS technology except the P6; some portions of the P6 use BiCMOS for added drive capabilities, and as thus, Intel refers to the process as a 0.6 micron BiCMOS process, despite all the CMOS parts being 0.5 microns wide. All use four layers of metal except for the K5, which makes do with only three. Despite relatively low supply voltages of 3.3 volts, all of the RISC processors dissipate 30W at normal operating frequencies, with the 21164 going through the roof at a whopping 50W power dissipation. The two x86 processors are less power hungry, with the K5 and P6 each dissipating approximately 20W. Their lower power consumption is due to the K5's fewer transistors and the P6's lower supply voltage (2.9V).

	AMD K5	21164	PA-8000	Intel P6	R10000	PPC 620	UltraSPARC
Transistor Count	4.3M	9.3M	~5M	5.5M	5.9M	6.9M	5.2M
Die Size (mm ²)	~250 - 300	298	~250 - 300	306	298	311	315
Power Dissipation	~20W	50W	~30W	20W	30W	30W	30W
Initial clock rate (MHz)	100	300	200	133	200	133	167
Supply voltage (V)	3.3	3.3	3.3	2.9	3.3	3.3	3.3
IC process	0.5 CMOS	0.5 CMOS	0.5 CMOS	0.6 BiCMOS	0.5 CMOS	0.5 CMOS	0.5 CMOS
Number of metal layers	3	4	4	4	4	4	4

Table 10: Physical characteristics

The number of transistors used for each processor is pretty close as well, with four processors hovering around the 5 - 6 million transistor count mark. Again, the 21164 wins this category, with an amazing 9.3 million transistor count; this is due primarily to the 96K on-chip L2 cache. The P6's transistor count does not include its L2 cache, as the L2 is on a separate die, but connected to the processor via a dedicated on-chip bus. The P6's 256K L2 cache itself eats up a whopping 15.5 million transistors, which explains why it was not placed on the same die as the CPU. (Note: the P6's L2 cache requires 15.5M transistors not only due to its size, but also because it is built using SRAM technology, which is inherently more dense than DRAM memory; each bit in the SRAM is built using six transistors.). On the lower boundary, the K5 uses only 4.3M transistors. While its execution and control engine is similar to the other processors, the K5 forgoes on-chip bus and L2 control logic, saving a large number of transistors. The transistor count of the PA-8000 is not known yet, but is estimated to be lower than the typical RISC processors, probably coming in around 5 million transistors. This is due to the PA-8000's total lack of on-chip cache; the number of transistors devoted to increasing the out-of-order window size do not take up as many transistors as even a decent size (16K or larger) unified on-chip cache.

Finally, the die sizes of each of the designs are surprisingly similar, all with areas within 5% of 300 mm². This means that a standard sized silicon wafer should be able to potentially turn out the same number of each processor. However, due to higher transistor counts, the yields of chips such as the 21164 and PowerPC 620 will likely be lower than their counterparts, as there are greater chances of defects with a larger number of transistors in a given die area.

15.0 Processor Performance

The performance of a processor is heavily dependent upon the system configuration and the application being run. However, since all of these processors, save the 21164, have yet to reach the market, our best estimate for performance is via the SPEC benchmark suite, which is the performance numbers that these companies choose to report. The following SPEC performance numbers are estimates taken from the first silicon samples:

	AMD K5	21164	PA-8000	Intel P6	R10000	PPC 620	UltraSPARC
SPECint92	147	330	>360	200	>300	225	275
SPECfp92	~75	500	>550	~200	>600	300	305

Table 11: SPEC92 benchmark performance

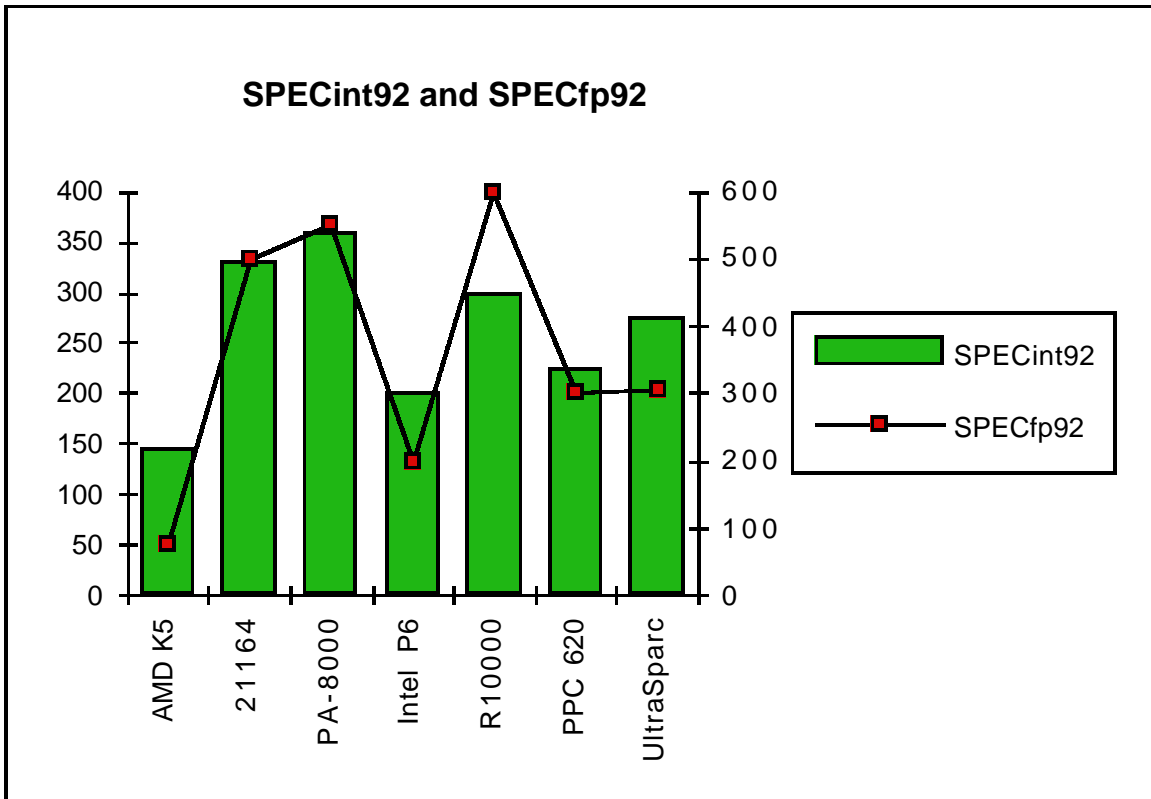


Figure 9: Relative SPECint92 and SPECfp92 performance

While SPEC92 performance may be indicative of a processor's potential performance on applications, it is more interesting to see a processor's efficiency. This is found by looking at a processor's SPECint92 performance compared to its clock rate. This ratio, SPECint92/MHz, gives a good approximation of the number of instructions executed per cycle (IPC). The SPEC/MHz ratio is a good relative performance for all these next generation processors, as each tries to attain the highest performance possible; none is building a processor which is only good when measured by SPEC/MHz, since real world performance is important for incorporation of these processors into future, high performance systems.

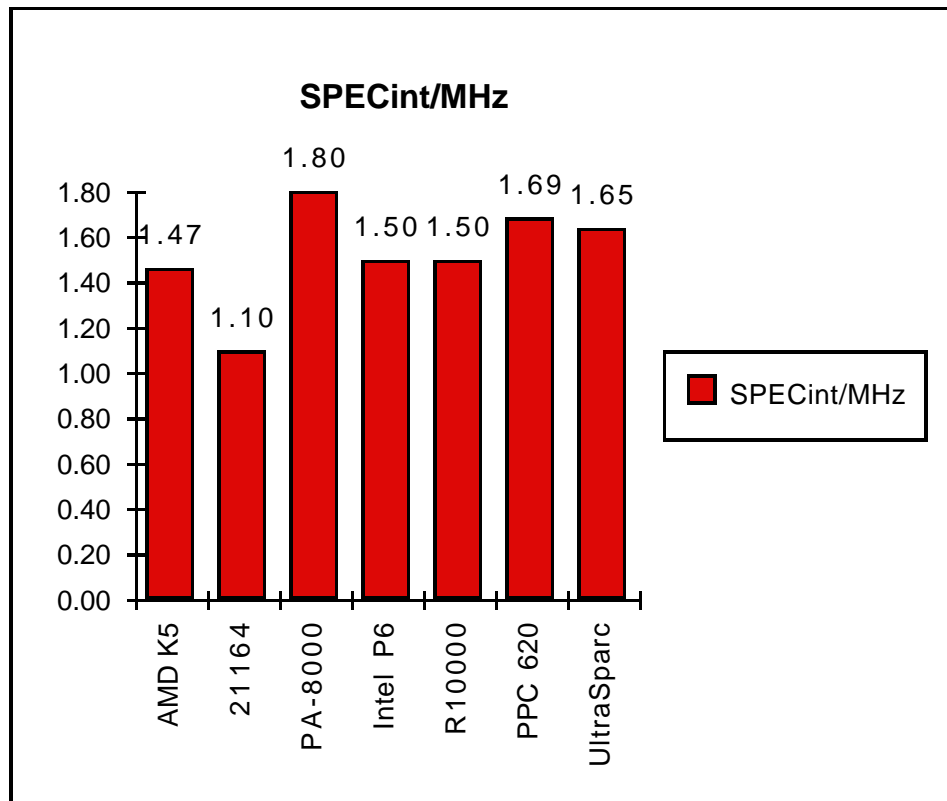


Figure 10: SPECint92/MHz comparison

All these processors achieve over 1 IPC performance. As expected, the PA-8000, with its 56 instruction out-of-order execution window, wins the race with an estimated 1.80 IPC. Close behind is the PowerPC 620 at 1.69 IPC, followed by the UltraSPARC, P6, and R10000. The K5 and P6 calculations are in terms of ROPs and uops, respectively, and not native x86 instructions. It is interesting to note that the 21164, which according to this metric is the least efficient processor, actually has the highest performance of the group. Though the PA-8000 is estimated to exceed the 21164 in SPECint92 performance, the PA-8000 is just beyond first silicon tests, whereas DEC already has shipping versions of the 300 MHz 21164.

As of now, the optimizing for clock rate ala the 21164 is delivering higher performance than the massively out-of-order engines of all the other RISC processors. While being extremely efficient (relatively) at 1.69 IPC, the PowerPC 620 delivers 225 SPECint92 at only 133 MHz. Via process upgrades and other minor tweaks, the PowerPC 620 is planned to reach 330 SPECint92 by the end of 1995 while running at approximately 200 MHz. However, that performance only matches that of today's currently shipping 21164. Processors like the R10000 and PA-80000 are narrowing the clock rate gap with respect to the 21164, with both expecting to operate at an initial frequency of 200 MHz. However, since there are no shipping parts to date, it remains to be seen whether these clock rates will actually be achieved. If they are not, their

claimed performance will drop considerably, magnifying the 21164's performance advantage. Thus, it seems that DEC did the right thing by, as Michael Shebanow noted in a guest lecture for EECS 598.3, by choosing to optimize speed over complexity whenever possible.

16.0 Conclusion

This report has presented a microarchitectural survey of seven of the recently announced "next generation" microprocessors. The processors presented were the AMD K5, the DEC Alpha AXP 21164, the HP PA-8000, the Intel P6, the MIPS R10000, the PowerPC 620, and the Sun UltraSPARC. Microarchitectural aspects of each microprocessor were compared, in an order akin to an instruction's path of execution through each processor. All the processors are multi-way, superscalar designs capable of achieving over 1 IPC of performance. Of course, this should be expected, as each processor attempts to dispatch at least three instructions each cycle. Most designers chose to implement complex, decoupled out-of-order execution architectures to achieve their high performance. However, the processor that actually has the highest performance, and is the only one that is currently shipping in volume, chose instead to optimize for clock speed. By choosing to reduce cycle time whenever possible, the DEC Alpha 21164 has a very low 1.1 IPC, yet attains the highest SPECint92 performance of all the processors. This is even more amazing when we remember that the 21164 can only dispatch four instructions per cycle for floating point code; for integer code, only two instructions can be dispatched per cycle. In effect, the 21164 is only a two-way superscalar machine for integer code, yet it achieves higher performance than slower clock rate machines that can dispatch up to four integer instructions each cycle. Thus, as we move on to higher and higher rates of dispatch, we must still be cognizant of the effects of complexity on clock rate and the tradeoffs that must be made. For now, optimizing for clock rate is winning over increasing complexity; however, should cycle times be reduced significantly for the complex designs, there is no doubt that they will be more successful in the future era of 8-, 16-, and higher-way superscalar designs.

Appendix:

Processor Quick Reference Sheets

Advanced Micro Devices K5

<i>Fetch Width per cycle:</i>	208 bits
<i>Number of predecode bits added per instruction:</i>	5
<i>Number of instructions dispatched per cycle (maximum):</i>	4
<i>Number of instructions retired per cycle:</i>	4
<i>Number of instruction in flight:</i>	16
<i>Pipeline depth (in stages):</i>	6
<i>Branch prediction method:</i>	Last time (1-bit)
<i>BHT/BTAC size:</i>	1024/none
<i>Estimated prediction accuracy (SPECint92):</i>	<80%
<i>L1 I-cache - Size, Associativity, Line Size:</i>	24K (16K effective), 4-way set associative, 16B
<i>L1 D-Cache - Size, Associativity, Line Size:</i>	8K, 4-way set associative, 16B
<i>Number of ways interleaved:</i>	4 way
<i>Non-blocking:</i>	no
<i>L2 Cache - Size, Associativity, Line Size:</i>	n/a
<i>On-chip/Off-chip, Interface:</i>	off-chip (requires external logic)
<i>Functional Units - Number, Type:</i>	6 total: 2 ALU, 2 L/S, 1 FPU, 1 Branch
<i>Reservation station/ROB/Instruction queue size:</i>	11
<i>Location:</i>	2 per ALU, 3 per L/S, 1 for FPU
<i>System Bus - Width, Protocol:</i>	n/a (requires external logic)
<i>Multiprocessor ready, number of processors, coherency:</i>	no
<i>Throughput:</i>	n/a
<i>Physical characteristics - Number of transistors, Die size:</i>	4.3M, ~250 - 300mm ²
<i>Supply voltage, Power dissipation, Initial clock rate:</i>	3.3V, ~20W, 100 MHz
<i>IC process, Number of metal layers:</i>	0.5 micron CMOS, 3 metal layers
<i>Estimated performance:</i>	147 SPECint92, ~75 SPECfp92

Digital Equipment Corporation Alpha AXP 21164

<i>Fetch Width per cycle:</i>	4 instructions
<i>Number of predecode bits added per instruction:</i>	5
<i>Number of instructions dispatched per cycle (maximum):</i>	4, 2 INT + 2 FP
<i>Number of instructions retired per cycle:</i>	4
<i>Number of instruction in flight:</i>	21 loads/6 stores
<i>Pipeline depth (in stages):</i>	7
<i>Branch prediction method:</i>	Smith
<i>BHT/BTAC size:</i>	2048/none
<i>Estimated prediction accuracy (SPECint92):</i>	80%
<i>L1 I-cache - Size, Associativity, Line Size:</i>	8K, direct-mapped, 32B
<i>L1 D-Cache - Size, Associativity, Line Size:</i>	8K, direct-mapped, 32B
<i>Number of ways interleaved:</i>	none
<i>Non-blocking:</i>	yes
<i>L2 Cache - Size, Associativity, Line Size:</i>	96K, 3-way set associative, 64B
<i>On-chip/Off-chip, Interface:</i>	on-chip
<i>Functional Units - Number, Type:</i>	5 total: 2 ALU, 1 L/S, 2 FPU
<i>Reservation station/ROB/Instruction queue size:</i>	none
<i>Location:</i>	n/a
<i>System Bus - Width, Protocol:</i>	128 bits, split-transaction
<i>Multiprocessor ready, number of processors, coherency:</i>	yes, 4 processors, MESI
<i>Throughput:</i>	1.6 GB/s
<i>Physical characteristics - Number of transistors, Die size:</i>	9.3M, 298mm ²
<i>Supply voltage, Power dissipation, Initial clock rate:</i>	3.3V, 50W, 300 MHz
<i>IC process, Number of metal layers:</i>	0.5 micron CMOS, 4 metal layers
<i>Estimated performance:</i>	330 SPECint92, 500 SPECfp92

Hewlett-Packard Precision Architecture 8000 (PA-8000)

<i>Fetch Width per cycle:</i>	4 instructions
<i>Number of predecode bits added per instruction:</i>	5
<i>Number of instructions dispatched per cycle (maximum):</i>	4, 2 compute + 2 memory
<i>Number of instructions retired per cycle:</i>	4
<i>Number of instruction in flight:</i>	56
<i>Pipeline depth (in stages):</i>	7 INT, 9 FPU/Memory
<i>Branch prediction method:</i>	Smith
<i>BHT/BTAC size:</i>	256/32
<i>Estimated prediction accuracy (SPECint92):</i>	80%
<i>L1 I-cache - Size, Associativity, Line Size:</i>	1M - 4M, direct-mapped, 64B (off-chip)
<i>L1 D-Cache - Size, Associativity, Line Size:</i>	1M - 4M, direct-mapped, 64B (off-chip)
<i>Number of ways interleaved:</i>	2-way
<i>Non-blocking:</i>	yes
<i>L2 Cache - Size, Associativity, Line Size:</i>	none
<i>On-chip/Off-chip, Interface:</i>	none (L1 caches are huge already)
<i>Functional Units - Number, Type:</i>	6 total: 2 ALU, 2 L/S, 2 FPU
<i>Reservation station/ROB/Instruction queue size:</i>	56 instruction queue: 28 compute, 28 memory
<i>Location:</i>	global
<i>System Bus - Width, Protocol:</i>	64 bit multiplexed, split-transaction
<i>Multiprocessor ready, number of processors, coherency:</i>	yes, 4 processors, MESI
<i>Throughput:</i>	960 MB/s
<i>Physical characteristics - Number of transistors, Die size:</i>	~5M, ~250 - 300mm ²
<i>Supply voltage, Power dissipation, Initial clock rate:</i>	3.3V, ~30W, 200 MHz
<i>IC process, Number of metal layers:</i>	0.5 micron CMOS, 4 metal layers
<i>Estimated performance:</i>	>360 SPECint92, >550 SPECfp92

Intel P6

<i>Fetch Width per cycle:</i>	128 bits
<i>Number of predecode bits added per instruction:</i>	none
<i>Number of instructions dispatched per cycle (maximum):</i>	3, 1 complex + 2 simple
<i>Number of instructions retired per cycle:</i>	3
<i>Number of instruction in flight:</i>	40
<i>Pipeline depth (in stages):</i>	12 compute, 14 memory
<i>Branch prediction method:</i>	Yeh
<i>BHT/BTAC size:</i>	512/512
<i>Estimated prediction accuracy (SPECint92):</i>	90%
<i>L1 I-cache - Size, Associativity, Line Size:</i>	8K, 4-way set associative, 32B
<i>L1 D-Cache - Size, Associativity, Line Size:</i>	8K, 2-way set associative, 32B
<i>Number of ways interleaved:</i>	4-way
<i>Non-blocking:</i>	yes
<i>L2 Cache - Size, Associativity, Line Size:</i>	256K, 4-way set associative, 32B
<i>On-chip/Off-chip, Interface:</i>	shares MCM with CPU - 128 bit direct bus
<i>Functional Units - Number, Type:</i>	6 total: 2 ALU, 3 L/S, 1 FPU
<i>Reservation station/ROB/Instruction queue size:</i>	20
<i>Location:</i>	global
<i>System Bus - Width, Protocol:</i>	64 bit, split-transaction
<i>Multiprocessor ready, number of processors, coherency:</i>	yes, 4 processors, MESI
<i>Throughput:</i>	528 MB/s
<i>Physical characteristics - Number of transistors, Die size:</i>	5.5M, 306mm ²
<i>Supply voltage, Power dissipation, Initial clock rate:</i>	2.9V, 20W, 133 MHz
<i>IC process, Number of metal layers:</i>	0.6 micron BiCMOS, 4 metal layers
<i>Estimated performance:</i>	200 SPECint92, ~200 SPECfp92

MIPS Technologies, Incorporated R10000

<i>Fetch Width per cycle:</i>	4 instructions
<i>Number of predecode bits added per instruction:</i>	4
<i>Number of instructions dispatched per cycle (maximum):</i>	4 (maximum of 1 store)
<i>Number of instructions retired per cycle:</i>	4
<i>Number of instruction in flight:</i>	32
<i>Pipeline depth (in stages):</i>	5 ALU, 6 Memory, 7 FPU
<i>Branch prediction method:</i>	Smith
<i>BHT/BTAC size:</i>	512/none
<i>Estimated prediction accuracy (SPECint92):</i>	80%
<i>L1 I-cache - Size, Associativity, Line Size:</i>	32K, 2-way set associative, 64B
<i>L1 D-Cache - Size, Associativity, Line Size:</i>	32K, 2-way set associative, 32B
<i>Number of ways interleaved:</i>	2-way
<i>Non-blocking:</i>	yes
<i>L2 Cache - Size, Associativity, Line Size:</i>	512K - 16M, 2-way set associative, 64B
<i>On-chip/Off-chip, Interface:</i>	Off-chip, on-chip control, 128 bit dedicated bus
<i>Functional Units - Number, Type:</i>	5 total: 2 ALU, 1 L/S, 2 FPU
<i>Reservation station/ROB/Instruction queue size:</i>	48
<i>Location:</i>	16 for integer ops, 16 for FP ops, 16 for memory ops
<i>System Bus - Width, Protocol:</i>	64 bit multiplexed, split-transaction
<i>Multiprocessor ready, number of processors, coherency:</i>	yes, 4 processors, MESI
<i>Throughput:</i>	1.6 GB/s
<i>Physical characteristics - Number of transistors, Die size:</i>	5.9M, 298mm ²
<i>Supply voltage, Power dissipation, Initial clock rate:</i>	3.3V, 30W, 200 MHz
<i>IC process, Number of metal layers:</i>	0.5 micron CMOS, 4 metal layers
<i>Estimated performance:</i>	>300 SPECint92, >600 SPECfp92

Apple/IBM/Motorola PowerPC 620

<i>Fetch Width per cycle:</i>	4 instructions
<i>Number of predecode bits added per instruction:</i>	7
<i>Number of instructions dispatched per cycle (maximum):</i>	4
<i>Number of instructions retired per cycle:</i>	4
<i>Number of instruction in flight:</i>	16
<i>Pipeline depth (in stages):</i>	5
<i>Branch prediction method:</i>	Smith
<i>BHT/BTAC size:</i>	256/2048
<i>Estimated prediction accuracy (SPECint92):</i>	90%
<i>L1 I-cache - Size, Associativity, Line Size:</i>	32K, 8-way set associative, 64B
<i>L1 D-Cache - Size, Associativity, Line Size:</i>	32K, 8-way set associative, 64B
<i>Number of ways interleaved:</i>	2-way
<i>Non-blocking:</i>	yes
<i>L2 Cache - Size, Associativity, Line Size:</i>	1M - 128M, direct-mapped, 64B
<i>On-chip/Off-chip, Interface:</i>	Off-chip, on-chip control, 128 bit dedicated bus
<i>Functional Units - Number, Type:</i>	6 total: 3 ALU, 1 L/S, 1 FPU, 1 Branch
<i>Reservation station/ROB/Instruction queue size:</i>	13
<i>Location:</i>	2 per ALU, 2 for FPU, 3 for Memory, 4 for branch
<i>System Bus - Width, Protocol:</i>	128 bit, split-transaction
<i>Multiprocessor ready, number of processors, coherency:</i>	yes, 4 processors, MESI
<i>Throughput:</i>	1067 MB/s
<i>Physical characteristics - Number of transistors, Die size:</i>	6.9M, 311mm ²
<i>Supply voltage, Power dissipation, Initial clock rate:</i>	3.3V, 30W, 133 MHz
<i>IC process, Number of metal layers:</i>	0.5 micron CMOS, 4 metal layers
<i>Estimated performance:</i>	225 SPECint92, 300 SPECfp92

Sun Microsystems UltraSPARC

<i>Fetch Width per cycle:</i>	4 instructions
<i>Number of predecode bits added per instruction:</i>	4
<i>Number of instructions dispatched per cycle (maximum):</i>	4 (4 th must be branch or FP)
<i>Number of instructions retired per cycle:</i>	4
<i>Number of instruction in flight:</i>	8 stores/9 loads
<i>Pipeline depth (in stages):</i>	9
<i>Branch prediction method:</i>	Smith
<i>BHT/BTAC size:</i>	512/2048
<i>Estimated prediction accuracy (SPECint92):</i>	88%
<i>L1 I-cache - Size, Associativity, Line Size:</i>	16K, 2-way set associative, 32B
<i>L1 D-Cache - Size, Associativity, Line Size:</i>	16K, direct-mapped, 32B
<i>Number of ways interleaved:</i>	none
<i>Non-blocking:</i>	yes
<i>L2 Cache - Size, Associativity, Line Size:</i>	1M - 128M, direct-mapped, 64B
<i>On-chip/Off-chip, Interface:</i>	Off-chip, on-chip control, 128 bit dedicated bus
<i>Functional Units - Number, Type:</i>	8 total: 2 ALU, 1 L/S, 3 FPU, 2 graphics
<i>Reservation station/ROB/Instruction queue size:</i>	12
<i>Location:</i>	Instruction queue between decode and FUs
<i>System Bus - Width, Protocol:</i>	128 bit, split-transaction
<i>Multiprocessor ready, number of processors, coherency:</i>	yes, 4 processors, MOESI
<i>Throughput:</i>	1.3 GB/s
<i>Physical characteristics - Number of transistors, Die size:</i>	5.2M, 315mm ²
<i>Supply voltage, Power dissipation, Initial clock rate:</i>	3.3V, 30W, 167 MHz
<i>IC process, Number of metal layers:</i>	0.5 micron CMOS, 4 metal layers
<i>Estimated performance:</i>	275 SPECint92, 305 SPECfp92

References

- [1] J. E. Smith, "A Study of Branch Prediction Strategies," Proceedings to the 8th annual International Symposium on Computer Architecture, 1981, pp. 135-148.
- [2] T.-Y. Yeh and Y. N. Patt, "Two-Level Adaptive Training Branch Prediction," 24th ACM/IEEE International Symposium in Microarchitecture, November, 1991. pp. 51-61.
- [3] C. V. Ravishankar and J. T. Goodman, "Cache Implementation for Multiple Processors," IEEE Comcon Digest, Spring, 1983. pp. 346-350.

AMD K5 references

T. R. Halfhill, "AMD vs. Superman." Byte Magazine, November, 1994. pp. 95-103.

M. Slater, "AMD's K5 Designed to Outrun Pentium," Microprocessor Report, October 24, 1994. pp. 1, 6-11.

DEC Alpha 21164 references

P. Bannon, "Micro-architecture of the Alpha 21164, the Second Generation Alpha Microprocessor," a talk given in EECS 598.3 on April 3, 1995.

D. Bhandarkar, "Internal Architecture of Alpha 21164 Microprocessor," 1995 IEEE Comcon Digest, pp. 79-87.

L. Gwennap, "Digital Leads the Pack with 21164," Microprocessor Report, September 12, 1994. pp. 1, 6-10.

"DEC Alpha 21164 Internal Architecture" class handout for EECS 598.3, February 13, 1995, pp. 1-1 to 1-42.

HP PA-8000 references

L. Gwennap, "PA-8000 Combines Complexity and Speed," Microprocessor Report, November 14, 1994. pp. 1, 6-9.

D. Hunt, "Advanced Performance Features of the 64-bit PA-8000," 1995 IEEE Comcon Digest, pp. 123-128.

Intel P6 references

R. P. Colwell and R. L. Steck, "A 0.6 micron BiCMOS Processor with Dynamic Execution," 1995 IEEE ISSCC conference, found on Intel's WWW site: <http://www.intel.com/procs/p6/proceed/proceed.html>.

L. Gwennap, "Intel's P6 Uses Decoupled Superscalar Design," Microprocessor Report, February 16, 1995. pp. 9-15.

T. R. Halfhill, "Intel's P6," Byte Magazine, April 1995. pp. 42-58.

Y. N. Patt, Class notes for EECS 598.3, March 8, 1995.

MIPS R10000 references

L. Gwennap, "MIPS R10000 Uses Decoupled Architecture," Microprocessor Report, October 24, 1994. pp. 18-22.

"MIPS R10000 Product Overview," October, 1994, found on MIPS' WWW site: http://www.mips.com/HTMLs/r10000_docs/r10000_Pr_info/R10000_Tech_Br_cv.html.

PowerPC 620 references

L. Gwennap, "620 Fills Out PowerPC Product Line," Microprocessor Report, October 24, 1994. pp. 12-16.

D. Levitan, T. Thomas, and P. Tu, "The PowerPC 620™ Microprocessor: A High Performance Superscalar RISC Microprocessor," 1995 IEEE Comcon Digest, pp. 285-291.

"Advance Information: PowerPC 620™ RISC Microprocessor Technical Summary," Motorola Order Number MPC620/D, October, 1994.

UltraSPARC references

D. Greenley et al., “UltraSPARC^(TM): The Next Generation Superscalar 64-bit SPARC,” 1995 IEEE Comcon Digest, pp. 442-451.

L. Gwennap, “UltraSPARC Unleashes SPARC Performance,” Microprocessor Report, October 3, 1994. pp. 1, 6-9.

“Introducing UltraSPARC: Designed For the Next Generation of Computing Needs,” September, 1994, found on Sun’s WWW site:
<http://www.sun.com/stb/Processors/UltraSPARC/WhitePapers/Intro2Sparc/introducing.html>.